Naval Ocean Systems Center
San Diego, CA
92152-5000

Technical Report 1436
August 1991

Command and Control Systems
Architecture and Engineering
Program Office

Battle Management Architecture Design Concepts

2

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152-5000

J. D. FONTANA, CAPT, USN
Commander

R. T. SHEARER, Acting
Technical Director

ADMINISTRATIVE INFORMATION

This work, accomplished during fiscal year 1991, was funded by the Naval Ocean Systems Center to support future Command and Control Department systems development, transition, and architectural integration. The effort described in this report was performed under the direction of Vic Monteleon and Chuck Mirabile, NOSC Code 405.

This work truly represents a group effort. It was authored by

- Vic Monteleon (Code 405)
- Chuck Mirabile (Code 405)
- Stan Connors (Code 432)
- Denny Mattison (Code 464)
- Dave Smith (Code 171)
- Mickey Vineberg (Code 171)
- Art Chagnon (Code 171)

and owes much of its substantitive content to the following major contributors:

- Mike Crowley (Code 41)
- Lorraine Duffy (Code 442)
- Bela Feher (Code 442
- Gil Myers (Code 41)
- Frank White (Code 421)

Released by
Vic Monteleon, Head
Command and Control Systems
Architecture and Engineering Program Office

Under authority of
R. C. Kolb, Head
Command and Control
Department

ACKNOWLEDGMENT

The authors would like to thank the excellent editorial work of Sid Miller and the outstanding cover graphics design of Holly Raley in helping to make this report possible.

EXECUTIVE SUMMARY

OBJECTIVE

To define an architecture for the development, operation, and modification of battle management systems. Battle management is broader in scope than Command and Control Communications and Intelligence (C^3I), providing the wherewithal for commanders to deploy and, if required, fight assigned forces effectively across the spectrum of conflict.

APPROACH

An open system design, driven by service requests, was selected and developed to take advantage of emergent technology, while providing a flexible and responsive system for both users and system engineers.

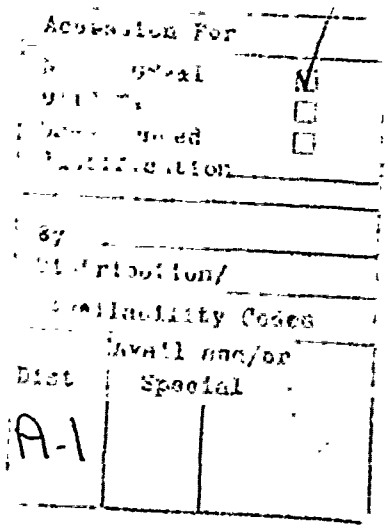
RESULTS

The concept was successfully applied and objectives were met. A detailed framework was provided for evaluation and system development.

RECOMMENDATIONS

1. Proceed into the next phase of development, test, and evaluation of the architecture.

2. Proceed with efforts to apply battle management architecture to current and developmental platforms and systems.



CONTENTS

FIGURES

INTRODUCTION

The Naval Ocean Systems Center (NOSC) initiated an effort in the fall of 1990 to produce a design concept for Battle Management Architecture (BMA). BMA has been defined in Ref. 1 as "a physical, functional, and organizational structure that supports command authorities from National Command Authority (NCA) through Element Commanders. It applies across the spectrum of conflict and enables commanders to employ and effectively control joint and combined forces, including naval forces, to achieve national objectives.... It includes and interrelates sensors, intelligence, command and control, combat direction, weapon system control, electronic warfare, and supporting logistic elements."

This report will present the NOSC formulation of (1) a framework for expressing the functions performed within BMA, (2) systems or implementations where BMA standards are required, and (3) attributes that characterize BMA. The effort currently is focused on developing guidelines for the design of systems within BMA and on a BMA implementation plan that includes concept validation, development support, accreditation, and transition from design to implementation. This current work will be reported separately.

When implemented, BMA will bring a standard approach to Navy battle management system development, operation, and modification. It will bring resource sharing, affording advantages of increased efficiency and survivability. It will also bring open systems that support: (1) the extensive use of commercial and commercially developed hardware and software, thereby improving performance and reducing costs and delays in acquiring new technology, (2) interoperability, and (3) the orderly evolution of our response to changes in threat and advances in technology.

Any architecture or system engineering effort for the U.S. Navy should address specific capability shortcomings of U.S. naval forces. It should also support independent platform, coordinated, and cooperative operations and must be compatible and interoperable during U.S. Navy, joint, and combined operations. A list of capability shortcomings of naval forces is provided as an appendix to this report. Once this list is complete, it can be related, through a matrix, to the attributes of the architecture or system engineering effort being proposed. This process should help in developing a meaningful set of needed attributes and should also be useful for deciding whether or not to continue an architecture or system engineering effort in a limited budget environment.

Results of the BMA Design Concept development effort are reported in the body of this document. First, the four-level Navy organization which BMA must support is described. Second, the candidate BMA concept is described. It consists of six layers, from functions (the user domain) to resources (the hardware). These six layers apply

to each command level of the Navy organization. The architecture is described in terms of the qualities, or attributes, which the architecture reflects, such as adaptability and flexibility, and the specific BMA features which supports each attribute. Guidelines for the development of systems under BMA are proposed. These guidelines include the development and use of BMA standards and support to system developers, including development and test facilities.

Since many new terms and phrases or old terms with new meanings are used in this report, a special list of definitions is provided as Appendix A.

NAVY ORGANIZATION SUPPORTED BY BMA

COMMAND LEVELS

The main objective of BMA is to provide the Navy with the methods and criteria to develop the necessary architecture to achieve the optimum warfighting capability in view of current and future threats (see Appendix B). This architecture will, of necessity, ensure mutual support among all levels of command. It will establish a consistent, coordinated direction for Navy programs, and it will ensure that warfare systems will be compatible with each other and mutually supportive.

The BMA structure is illustrated in Fig. 1. The traditional levels of command—NCA; unified/specified component, and Fleet commands; force and warfare commands; and platform/unit commands—perform similar functions during certain levels of conflict. When these levels of command are combined to eliminate unnecessary redundancy, a minimum representative set of command levels is derived: the BMA levels of command shown in Fig. 1 as Global, Theater, Force, and Platform. Information, acquisition, and control flow along connectivity paths, shown by the solid lines in Fig. 1. The command levels represent the organizational structure of BMA; the nodes and the connectivity between them represent the functional structure; and the resource elements, which are created from building blocks, represent the physical structure of BMA.

NODES

Nodes (also called *Command Nodes*) are the underpinnings of the BMA organization. A node (1) embodies all the physical aspects of the command, including the commander, related to battle management, (2) is the physical construct in which the command process is performed, and (3) performs a set of functions (Ref. 1). The BMA comprises Global, Theater, Force, and Platform nodes. Nodes can exist at any of the four levels. A ship—including its commanding officer, its crew, and all its assets—is an example of a platform-level node, a battle force—including its Officer in Tactical Command (OTC) and all its platforms—is an example of a force-level node. Each node imposes functional requirements on lower level nodes.

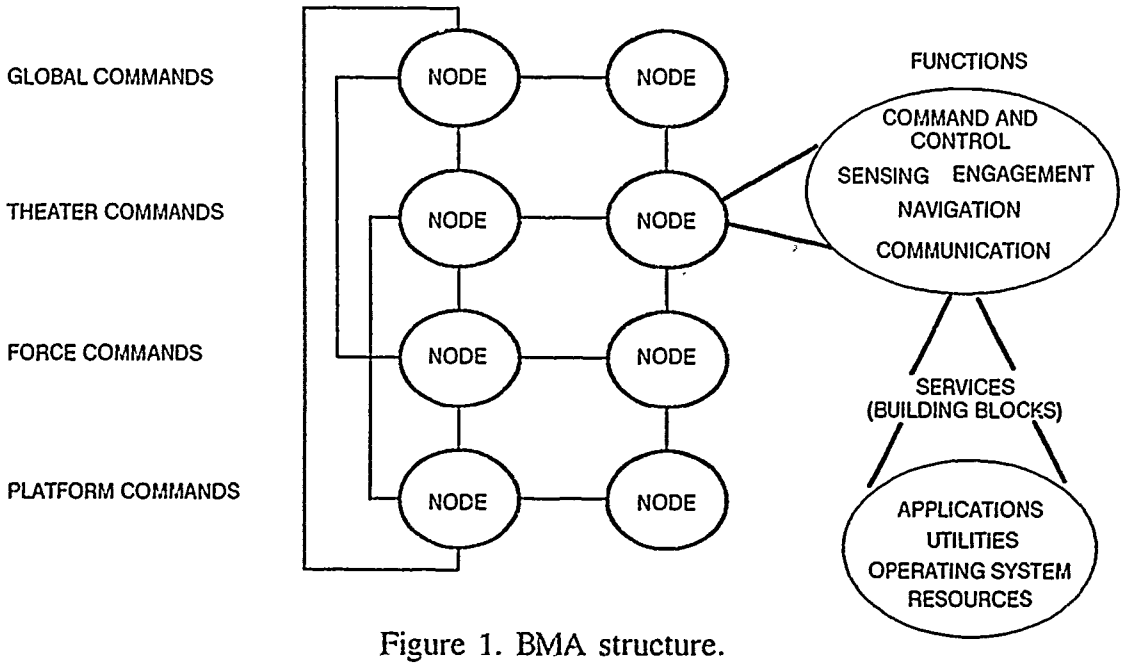


Figure 1. BMA structure.

Figure 1 shows that BMA nodes are connected between command levels and within command levels. The key for the future is the ability of BMA to support multiple levels of command. The current hierarchical organization may or may not endure; the concept of a command hierarchy certainly will.

Most functions will be implemented at platform nodes (ships, planes, submarines, etc.), even those which support higher-level nodes. Nodes are viewed here as a nested hierarchy where each node above the platform level comprises one or more lower level nodes and where each node below the global level inherits requirements from the higher level nodes; for example, a platform must support the force.

Functions and building blocks, which reside at the nodes, are the constituent elements or the BMA structure and are developed more fully later in this report.

ECHELON SKIPPING AND RECONSTRUCTION

BMA will apply throughout the command structure. This structure can be reconfigured to respond to contingency requirements. The chain of command orders the flow of information, under peacetime conditions, from NCA, through the theater and force commanders, to the platform commander.

During critical operations, an echelon commander may skip one or more echelons between him and the intended recipient of the command directive. This process, referred to as *skip echelon*, might reflect an operational necessity (to ensure a timely response) or the management style of the echelon commander.

Each echelon within the physical command structure represents a discrete command node, which is both a supplier and user of information contained within the overall command and control system. The dynamic nature of the system supporting the skipping of echelons is also reflected in the occupancy at each echelon. The loss of a current occupant of one of the echelons normally results in replacement from the next lower level and migration from the bottom up. Except for the highest level of command (NCA), whose replacement is covered by other procedures, lower echelon occupants must be capable of assuming and prepared to assume the functions of the next higher level of command. As an example, following the attrition of the force level command node, one of the platform echelon members must fill the vacuum at the force level.

BMA STRUCTURE AND ATTRIBUTES

LAYERED OPEN SYSTEM ARCHITECTURE

BMA is an open architecture, layered as shown in Fig. 2. The open architecture is partitioned vertically into layers so that each layer supports the next higher layer. An *open system* is a layered architecture whose interfaces are defined by industry standards, controlled by recognized standards organizations (such as ISO, ANSI, IEEE, and SAE).* Open architecture thus supports the rapid integration of commercial technology.

Layering permits independent (concurrent) work within separate layers and enables independent evolution within the layers. Appendix C contains two examples of layering: the IBM System/360 and the Open Systems Interconnection (OSI) Reference Model.

Throughout the following discussion of the open architecture layers, the term *user* indicates a commanding officer of a node or a subordinate authorized by the commanding officer to interact with BMA.

Functions

Functions are what a user does. To perform functions, a user generates *commands*, requests for system services which go to the next lower layer. Example sets of functions are listed in Appendix D.

---

* ISO International Standards Organization
ANSI American National Standards Institute
IEEE Institute of Electrical and Electronic Engineers
SAE Society of American Engineers

Although the layered architecture permits a modification of the functions, the manner in which functions are supported by the system is expected to be stable. A hierarchy of functions can be expected to achieve stability when the hierarchy consists of stable subassemblies (Ref. 2). Systems within the architecture must be capable of operating in an environment where the operational requirements change rapidly. A *subsumption architecture* supports this environment and is used to describe the functional BMA.

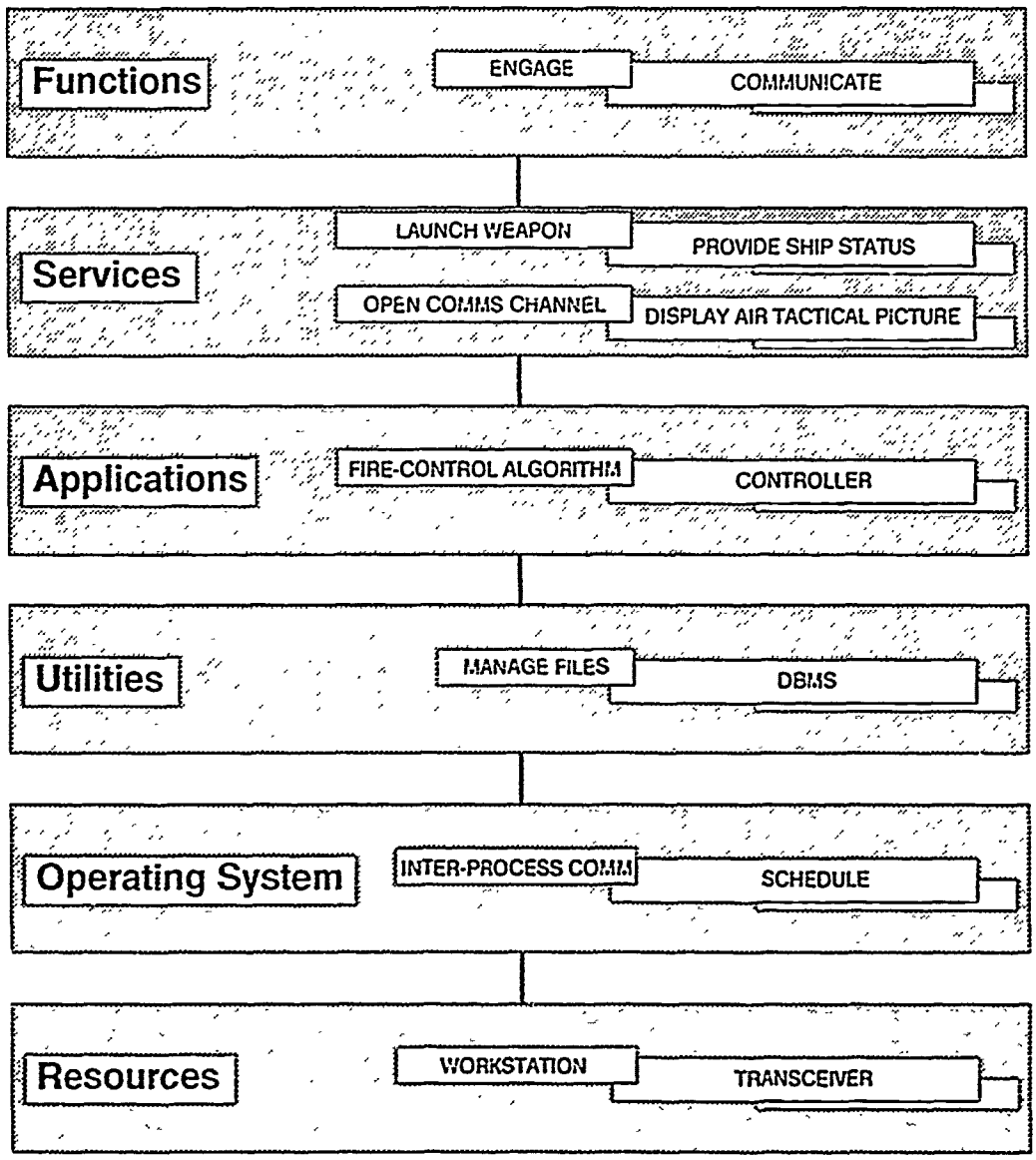


Figure 2. Open system architecture.

The concept of a subsumption architecture, and its application as a control mechanism in battle management, is straightforward. Subsumption architecture has been

applied in robotics to mimic very complicated behavior patterns by using a series of less complicated behaviors (referred to as "building blocks").

The similarity between robotics and naval warfare might not be immediately apparent. The moving robot operates in an environment where it must sense objects around it and make rapid decisions (or modify its behavior) based upon the nature of those objects. The individual platform, in naval warfare, has a similar objective, that of moving and rapidly responding to the nature of objects that are sensed (threat platforms or friendly units). For both the robot and the platform, each of the behaviors that must be accomplished can be divided into individual tasks that differ in complexity. A subsumption architecture is task-behavior oriented, rather than oriented around primary functions (Ref. 3). A primary function could be comprised of several behavioral tasks. Each specific behavioral task becomes a discrete layer of a building block structure, which is a ranked hierarchy according to its "level of competence" or complexity level. The entire control system is comprised of these "competence layers," which are combined to perform more complicated functions.

Layers of Control. A subsumption architecture is constructed by first creating a control system to achieve the lowest level of performance required. For a ship, this *zeroth level* allows the ship to function as a ship and accomplish such fundamental tasks as going to sea, maneuvering, navigating, and communicating.

The *first-level* control system is capable of examining data from the zeroth level and injecting data into the internal interfaces of the zeroth level in order to alter the behavior of that level. The first-level control system might correspond to surveillance tasks. The same process is repeated to achieve successively higher levels of competence.

The subsumption architecture is depicted in Fig. 3. Example sets of functions are listed in Appendix D. A refined set of functions, under development, is outside the scope of this document.

In Fig. 3, level 0 operations are the basic functions necessary to navigate and operate the platform. Level 1 is a function of higher complexity, that of searching for and sensing objects that are within the view of the platform. Level 2 includes the characterization and grouping of sensed objects into tracks, which is an operation of higher complexity than levels 0 or 1. Level 3 includes the individual platform engagement, or the enabling and placement of weapons on tracks.

There is a specific distinction between the functions described as Level 4 (coordinated engagement) and Level 5 (cooperative engagement). Coordinated engagements can be viewed as controlled individual engagements, where the controls might consist of assigning weapons to each participant in the coordinated engagement. Cooperative engagements, however, might involve the launch of a weapon by one platform and the

control of the weapon by another platform. The cooperative engagement is, therefore, the most complex function provided in the subsumption architecture. The arrows in Fig. 3, extending from more complex to less complex functions, indicate a feedback loop. Performance of functions at higher levels of complexity have implications for functions having a lower level of complexity, and might require changes in specific actions at lower levels.

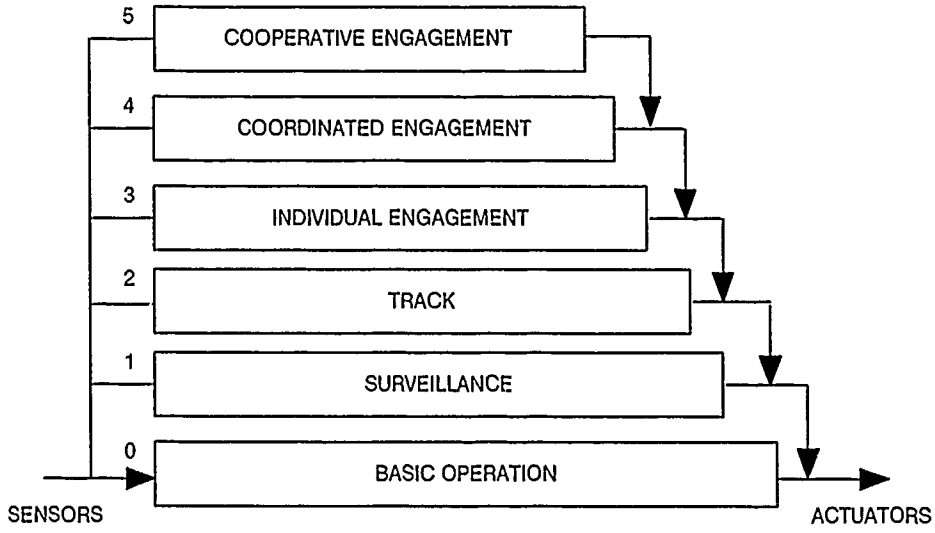


Figure 3. Hierarchical levels of control in a subsumption architecture.

Elements of the Levels. Each level in the architecture exists as a *finite-state process* (a process which, at any time, (1) is in one of a finite number of states, (2) may accept one of a finite number of inputs, (3) enters a new state—or remains in the current state—based on the current state and the input, and (4) produces an output according to the new state). Each state for a specific level carries the name of a primitive of a task to be controlled within that level. Each primitive has an associated command and is assigned a priority, as explained below.

Primitives and Commands. A *primitive* is a basic unit of performance; a more complex behavior can be derived by combining primitives within a level of control. A primitive at one level may be decomposed into primitives at the next lower level.

Commands are the means of accomplishing a primitive. The set of commands can be viewed as a command language into which user functions are mapped. A command is a request to the services layer for a system service. Each active command must have a unique priority to prevent system deadlock (see system services).

The process of decomposing primitives into commands is under study. An important question is when to decompose higher level primitives into lower level primitives

within the functions layer, yielding lower level commands, and when to pass high-level commands to the services layer for decomposition there. Support for both methods will allow the user to submit low-level commands when resources are not available to comply with an equivalent high-level command (see system services).

There are analogies in this decomposition problem to the design of computer assembly languages to support high-level languages; in each case, convenience of representation at the higher level, efficiency of translation from the higher to the lower level, and efficiency of implementation at the lower level must be considered.

Priorities. A priority is assigned to each primitive according to predetermined criteria. Default priorities may be established based on the operational plan. Priorities are inherited by the commands which accomplish the primitives and used within the services layer to resolve resource contention.

To prevent deadlock, no two active commands may have identical priorities. The system can enforce this design rule by using the lowest order field of the priority word to distinguish command priorities when necessary. This field can also be used to assure that no two nodes assign identical priorities by using a randomization algorithm.

Priorities determine which commands receive attention and are a powerful instrument of command policy. Therefore, it is important to establish clear policies governing the assignment of priorities. The architecture will support the establishment of priority ranges consistent with command doctrine.

Segmentation and Delegation of User Capabilities. Each node has at least one user, the commanding officer. He may authorize additional (subordinate) users. For example, an OTC might delegate authority for ASW operations to an ASWC. He may delegate authority in two ways. First, he may segment his command repertoire into subsets sufficient to support subordinate users in given areas. Subsets may vary in size and may or may not overlap. Second, he may authorize the use of all or part of this priority range by subordinate users. By delegating less than the full priority range, the user reserves the ability to make commands of higher priority than any by subordinate users.

Command and Control Interface. The Command and Control (C^2) interface provides the following capabilities in direct support of the user:

1. Translation of functions into commands.
2. Assignment of priorities to commands.
3. Partitioning of command repertoires into subsets (under such categories as "ASW" and "AAW").

4. Delegation of command subsets to subordinate users.

5. Delegation of user priority range subsets to subordinate users.

Services

Services are what the system does. A user, through commands, tells the system what to do; the system determines how to do it. Services are also event driven and can be driven, for example, by a sensor, weapon, or the platform itself. There are two classes of services, system services and element services. *System services*, performed in response to commands, or events, are higher level services. Responses are information returned to the functions layer in response to commands. The set of system services has been partitioned into elements according to the functions they support. *Element* services, the lower level services, are the mechanism by which elements support system services. An element service is supplied to the decision support element (responsible for system services) by another element in response to a *service request*. Figure 4 shows the processing hierarchy and associated hardware required to interface service requests from the commander/user into the physical architecture. Here, the commander/user enters into the system a command which then becomes a service request. The service request is then decoded and ranked relative to other service requests entering the system. After the service request is ranked, its execution is scheduled according to its priority and the ability of the systems within the architecture to respond. Example sets of commands, services requests, and services are listed in Appendix E. Within this layer, priority-based scheduling and resource allocation are performed.

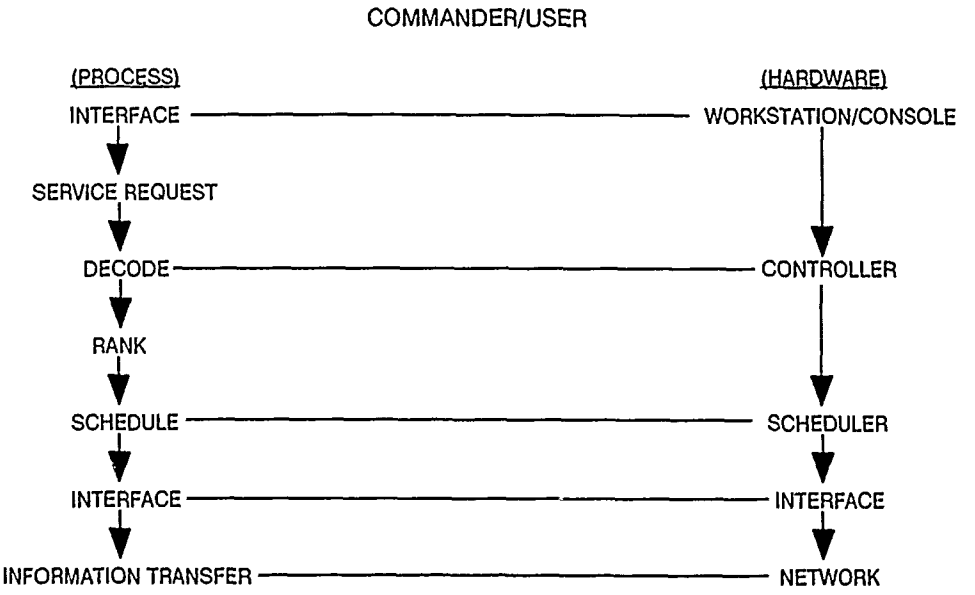


Figure 4. Service request processing.

System Services. *System services*, as has been said, are performed in response to commands (specifically, from the functions layer). For each command, there is a corresponding system service. A fundamental BMA design rule is that no system service can cause a deadlock (see Deadlock Prevention in Appendix F).

The set of system services is not "complete." On the contrary, the set is extensible. Even if it were possible to define the set perfectly now, it would require future change. Once a fully supported version of the command language is operational, care will be required to assure upward compatibility in future versions.

Elements and Element Services. An *element* is a (functional) entity which provides a set of *services*. Services are divided among elements according to the functions they support. The decision support element provides *system services* in response to *commands* from the functions layer. Other elements provide *element* services in response to *service requests* from the decision support element.

The service structure is shown in Fig. 5. The decision support element provides an interface which accepts commands, provides system services, and returns responses. It also performs priority-based scheduling and resource allocation so that no system service causes a deadlock. The decision support element may make *requests* (for service) to other elements and receive *replies*. *Element services*, the lower level services, are the mechanism by which those elements support the decision support element, i.e., respond to service requests. For each service request, there is a corresponding element service.

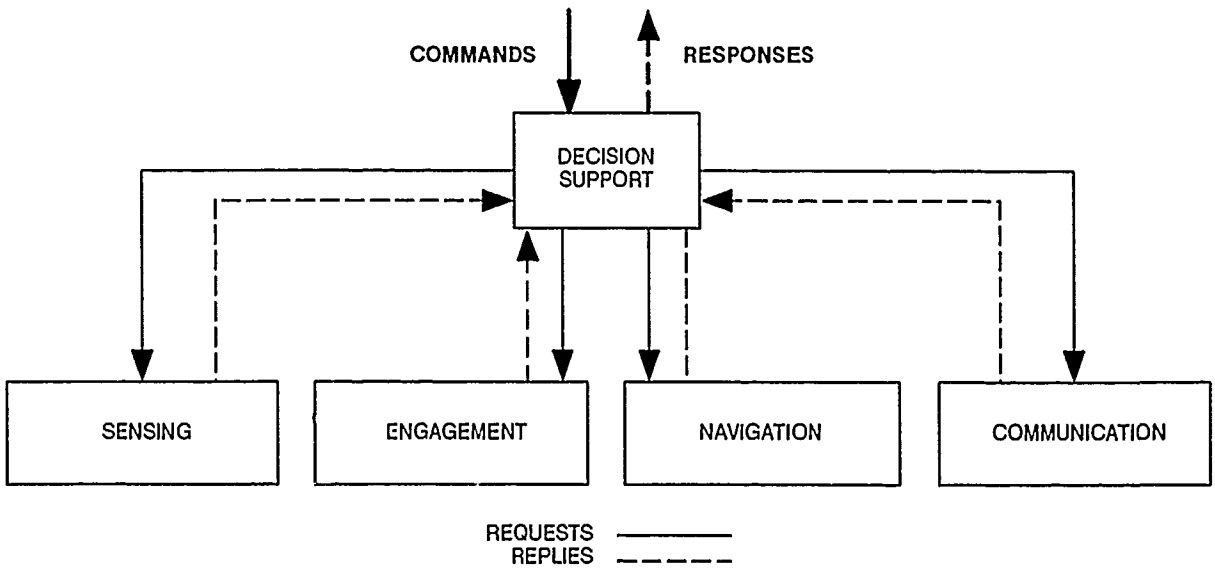


Figure 5. Service structure.

An element is capable of controlling resources. As mentioned under functions, command priority is inherited by all supporting services and then used to resolve contention for resources. Element configurations vary with respect to the services they are performing. Each element has an active controller (discussed under applications).

Service Session. A *service session* is the time during which an element performs a service. An element controls resources from the beginning of a service session until it no longer needs them to perform the service. A service session will be interrupted by the element controller if a needed resource is preempted. *Preemption* is the withdrawal of access to a resource, generally in response to the need for that resource to support a higher priority service.

Element Service Hierarchy. Node hierarchy is preserved within the service structure. A decision support element at a node issues requests to elements within the same node; it issues commands to decision support elements at lower levels. Note that this effectively extends the definition of a *command*. Figure 6 shows an example of the flow of commands and responses between a node (at Force Level) and a node at the next lower (Platform) level. These exchanges rely on communication element support.

Applications

Applications are (software) processes which support services. Examples of applications are signal-processing algorithms and decision aids. Elements, introduced in the services layer, are useful here for grouping and explaining applications.

Element Controller. One application common to all elements is an *element controller*. The *controller* accepts commands or requests, controls service sessions, and makes responses or replies. To accomplish these tasks, a controller is activated (normally on a workstation) after initialization and remains active for each element at each node throughout system operation.

The decision support element controller performs the following tasks to support a system service:

1. accepts commands
2. determines which element services are required to perform a system service
3. ascertains (from other elements) whether resources are available to provide needed services (see Deadlock Prevention in Appendix F)
4. provides responses (acknowledgments, etc.)
5. controls system service sessions

6. requests element services
7. provides certain general services

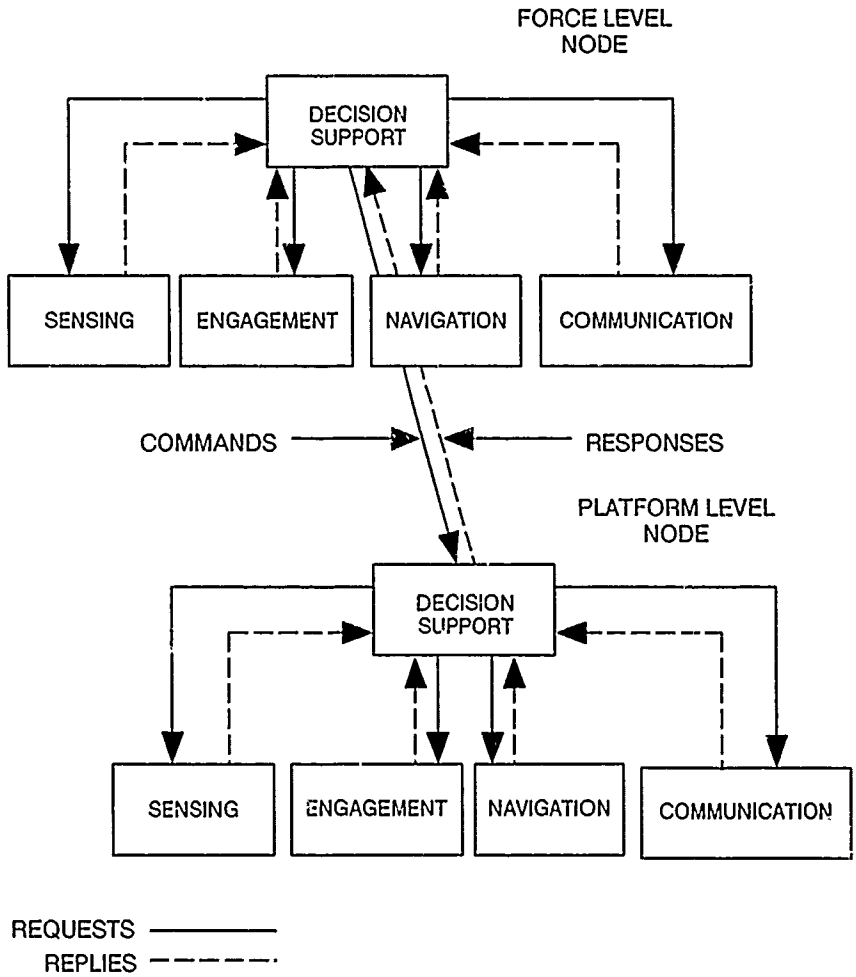


Figure 6. Node hierarchy within the service structure.

An element controller performs the following tasks to support the decision support element:

1. alerts the decision support element controller as to whether or not it can gain access to the resources required to provide a given service at a given priority (resources are shared when necessary and possible)
2. accepts requests
3. allocates resources to services
4. sends replies to the decision support element

5. controls element service sessions

6. deallocates resources

Decision Support Element. The decision support element supports command planning and decision making by means of assessment and evaluation. A variety of applications such as decision aids, tracking algorithms, and planning aids, will be available.

In addition, two applications necessary to support readiness reporting belong here. These support tactical (operational) and technical (maintenance) users. The two applications will synthesize status reports received from resources in order to support services reporting operational capability to tactical users and maintenance requirements to technical users.

Sensing Element. The sensing element detects and characterizes objects and characterizes the environment external to the node. The sensing element depicted in Fig. 7 accommodates sensors with various levels of signal processing capability. Assume that a full signal-processing capability can be partitioned into three stages to facilitate definition and specification. The information flow in Fig. 8 handles sensors with one stage, two stages, all three stages, or no signal processing capability, by allowing the missing stages to be provided external to the sensor. This is accomplished by defining standard interfaces at each stage. Once signal processing is complete, information can be passed through for similar-source and multisource integration (under a decision support application).

The inputs to the similar source integration boxes in Fig. 8 were grouped so that the sensors' outputs are easily combined by association, correlation, or fusion. Search radar contacts and detections from multifunction radars are of the same type and are easily compared or combined. Efforts are currently underway to relate electronic support measure (ESM) results to sensor radar output; therefore, ESM was combined with the radar outputs.

Acoustic sensors (active and passive sonars) were considered similar because they occupy the same portion of the frequency spectrum. Infrared, electro-optical, and laser outputs were also combined in one similar source integrator because of their frequency relationship within the spectrum (possible frequency/wavelength processing considerations).

If it becomes possible to further combine sensor outputs for similar source integration in the future as a result of advances in processing technology, this combination should certainly be done.

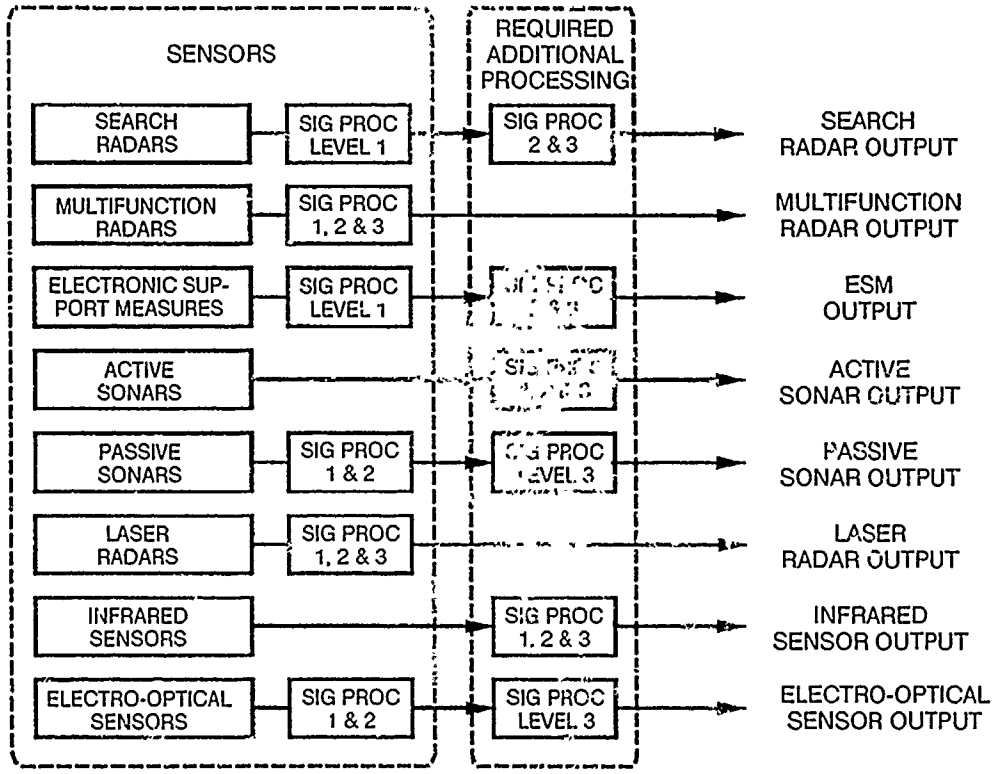


Figure 7. Sensing element application relationships.

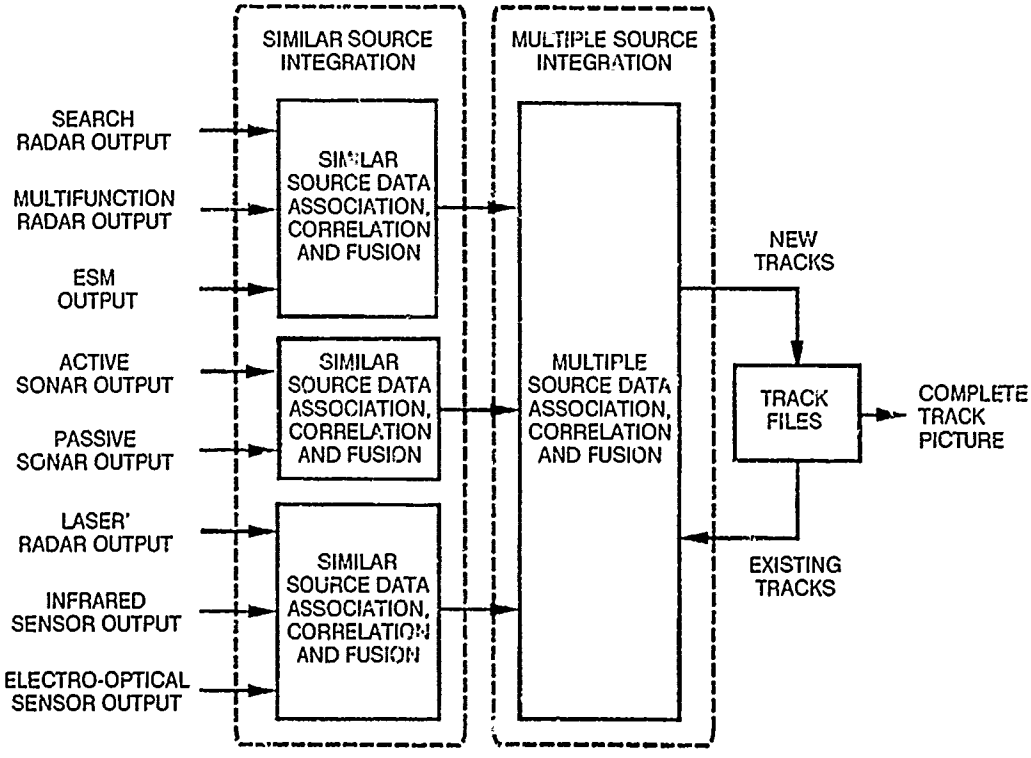


Figure 8. Similar and multiple source data applications.

Engagement Element. The engagement element must accommodate devices with various stages of capability and weapons with various stages of required preparation. Figure 9 shows a hypothetical flow diagram for an engagement element.

Navigation Element. The navigation element provides services in either an automatic or manual mode. In the automatic mode, orders are sent to the propulsion and radar controls; in the manual mode, they are not. Otherwise, the services are identical. In either case, applications are needed to supply supporting information (such as maps or tracks), to record all navigation modifications, and to support mode switching.

Applications will be required to support Navy interaction with Long-Range Navigation (LORAN) and the Global Positioning System (GPS). Differential GPS, which supports the calibration of local equipment with respect to known reference points, will become essential to navigation accuracy and will require support. Care will be required to assure that Navy requirements are met with the framework of emerging efforts to "internationalize" radio-navigation systems. In other words, monitoring emerging standards will be the key to profiting from growing worldwide GPS and LORAN support.

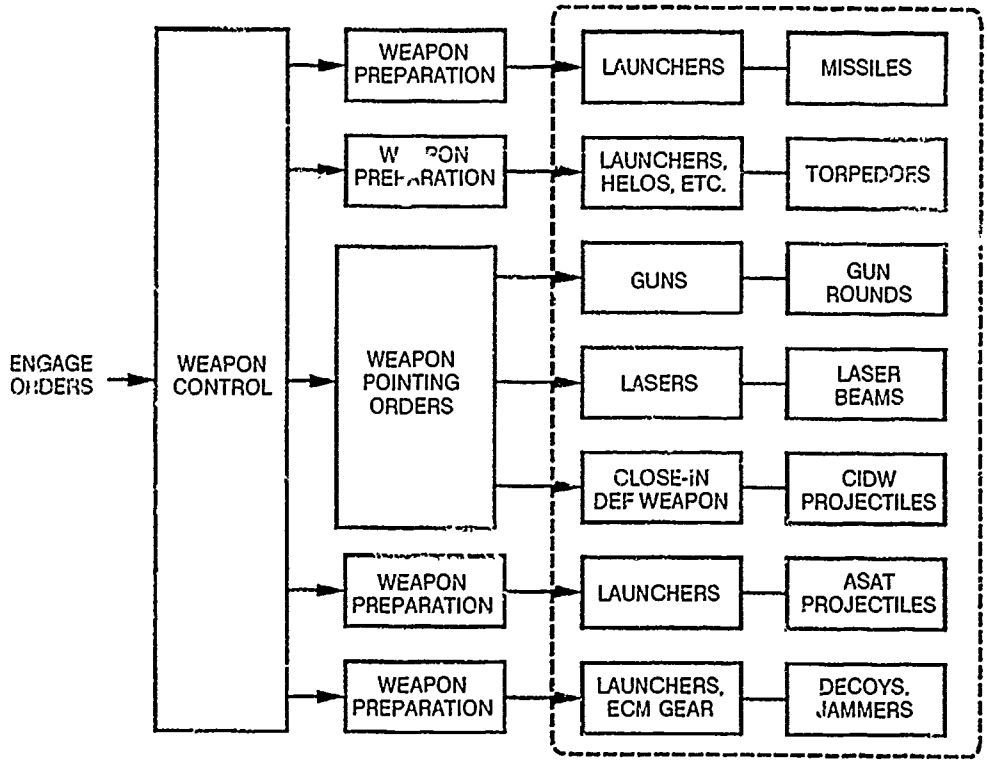


Figure 9. Application relationships of platform engagement element.

Communication Element. The communication element provides intra-node communication at the force level and higher and internode communication at all levels (intranode communication at the platform level is provided by the platform data distribution system and the operating system). For purposes of BMA, the communication element must include all seven layers of the International Standards Organization (ISO) Open Systems Interconnect (OSI) Reference Model (described in Appendix C). The entire OSI Reference Model is service oriented; each layer supports the next higher layer.

NOSC, under the direction of SPAWAR PD-50, has begun the development of the Communication Support System (CSS), a proposed Navy communications subarchitecture (Ref. 4). Since, as currently envisioned, CSS supports the concepts of services and shared resources, it is a candidate communication element for BMA. Of course, any BMA element must be designed and implemented as an open system, and BMA communications, by definition, are virtual.

Utilities

Utilities are the general-purpose processes necessary to support a broad range of applications. The utilities discussed below are representative and by no means constitute a comprehensive set.

File Management. The file server, a general-purpose utility, is of value to BMA because (1) it enables the retention of as many files as can be stored in a mass memory (all the applications or other utilities that might ever be used at a node) and their dissemination on demand wherever needed; (2) it supports rapid system reconfiguration for whatever reason; and (3) it provides backup for working copies of applications and utilities.

Database Management. The database management system (DBMS) supports the storage, update, correlation, and retrieval of data for applications. It is the mechanism which enforces common data formats, thereby supporting interoperability. While DBMS technology is mature, DBMS technology for battle management is not.

Commonly used techniques such as voting and time-stamping, which support concurrence in distributed DBMSs, must be flexible enough to support individual node initiatives in the battle managment application. This is necessary to address *network partitions* (situations when one or more nodes on a network cannot communicate with one or more other nodes on the network) or response time constraints. Layered DBMSs (constructed from multiple schemas, like layered communication systems), which are adaptable and flexible, are recommended for use in BMA.

Network Management. Management of the data distribution system includes a variety of tasks, such as initialization, token creation and recreation, fault detection,

reconfiguration, and subscriber entry and exit. These are common tasks of commercially available network management systems.

Support Processing. It will at times be necessary to perform a wide variety of support processing. Graphics processing will be needed to create, display, store, update, and transmit graphic images. Word processing will be needed to create, display, store, update, and transmit text files. Image processing will be needed to receive, scale, display, analyze, store, and transmit graphic images. I/O processing will be needed to receive and transmit files on the LAN. In addition, certain peripherals will be dedicated to a workstation and require special handlers.

Operating System

The operating system provides support to make the resources usable by the utilities and applications. The distributed operating system allows distributed resources to function as an integrated system. In a distributed operating system, each resource hosts a local version of the operating system, allowing it to send information to and receive information from other parts of the operating system in order to accomplish the following tasks:

- Schedule intraresource operations
- Allocate and deallocate low-level resources (peripherals)
- Allocate and deallocate main, secondary, and archival memory
- Manage interprocess communication using LAN protocols
- Manage intraplatform circuit switching using LAN protocols
- Monitor and measure performance
- Maintain logs, system state, and backup information
- Manage system initialization and recovery (warm and cold system start)
- Manage intraplatform electronic mail
- Maintain user profiles
- Maintain user mailboxes and supply them when and where needed
- Initiate user sessions using login procedures, account and password verification, and user profiles
- Terminate user sessions using logout procedures, saving files and any user profile information.

Resources

A *resource* is a hardware device, either special-purpose (sensor, weapon, etc.) or general-purpose (local area network, workstation, etc.), necessary to support processes (via the operating system). Each resource records information necessary for resource allocation. A resource, for example an antenna, may be shared; that is, it may support more than one process at a time. Resource allocation is discussed in Appendix F.

General-Purpose Resources. A general-purpose resource is one which generally supports a broad range of applications and utilities. The following are examples: local area networks (LANs); workstations which include a backplane, a memory, a CPU, a monitor, and a LAN interface; mass storage devices; file servers; and signal processors. The set of general-purpose resources discussed below is representative:

Local Area Network (LAN). A LAN is a cable, probably fiber-optic for near-future implementations, with taps allowing "plug-in" connectivity. It provides control and data connectivity among all platform resources and users and is the heart of the platform resource layer. It supports interoperability by requiring resources to connect to it through standard physical interfaces and to use it through standard communication protocols. It provides adaptability by allowing new resources to be added with relative ease. It significantly reduces cabling from today's point-to-point shipboard requirements. Finally, it supports resource sharing by substituting "horizontal" connectivity for "vertical" connectivity. This "heart of the platform resource layer" must be survivable. It must operate as specified even in the presence of damage. The Survivable Adaptable Fiber Optic Embedded Network II (SAFENET) is a candidate for shipboard LANs (Ref. 5). It is designed as an open system, is based on industry standards, and is designed to be survivable. Survivability is based on a counterrotating token ring design which tolerates a variety of fault conditions, lends itself to rapid repair, and allows for the distribution of the two rings of the token ring network.

Workstation. A workstation is a device, generally including a backplane, a memory, a CPU, a monitor, and a LAN interface; it is designed to connect to a network and to provide both high-power computing and local control and access. In the commercial sector, workstation performance has been upgraded at regular intervals so that new workstations offer high-speed computing and extensive, expandable storage. Workstation designs generally allow users to benefit from this trend through standard backplanes that allow less capable workstations to be upgraded through the addition of new cards (carrying CPU chips, memory, etc.). Workstations on a network can be used as remote computing resources or can exploit other remote resources for computing and data and file access. There may be many workstations on a platform, each supporting various applications and utilities. Programmable display configurations will allow a workstation to adapt to user needs when those needs arise.

High-Speed Data Transfer Network. A high-speed data transfer network is made up of cables, probably fiber-optic for near-future implementations, and a logical device which controls how the cables are connected; it is capable of interconnecting two resources for the purpose of high-speed data transfer. Such a network enables the decoupling of BMA resources, for example a sensor and a signal processor, so that they can be connected in various configurations and shared as needed. This will decrease the cost of procuring resources, which need no longer be vertically integrated, and increase survivability by replacing series resource configurations with parallel resource configurations. High-speed data transfer networks must be survivable. Simple redundancy and the availability of spare components ("hot and cold sparing") will probably be adequate to protect switch operations during combat, but the technology is not yet available.

Mass Memory. A mass memory is a device capable of storing very large amounts of data. Mass memories provide the hardware technology to implement file management (file servers) and database management.

Signal Processor. Although listed here, the signal processor actually falls into a broader category of "special processors"; a special processor is designed for a certain class of applications. Other such processors include backend database processors, vector and parallel computers, associative processors, and neural networks. While this category will not die out, the remarkable performance progress of workstations will continue to change its application space. The workstations of today can take on much of the load borne by yesterday's signal processors.

Metropolitan Area Network (MAN). A MAN is a backbone network, interconnecting local area networks. A MAN may be included at a major shore site or aboard a very large platform.

Special-Purpose Resources. A special-purpose resource is one which directly supports particular element services (see Appendix G). Sensors, weapons, navigation aids, and communication devices are examples. The availability of general-purpose resources and the possibility of interconnecting a special-purpose resource (e.g., a sensor) to a general-purpose resource (e.g., a processor) via a switch is expected to simplify special-purpose resources. As implied in the discussion under Applications, standards are required to accomplish this simplification.

Resource Interconnection. In general, all resources at a platform node are connected by one or more LANs. Interconnections for high-speed data transfer may be accomplished by using one or more high-speed switches. The switch is also connected to and controlled via the LAN.

Service-Defined Element Configurations. An element may use both special-purpose and general-purpose resources to provide a service; it will generally control a resource

only as long as needed to provide that service. Examples of service sessions in Appendix H illustrate how elements are delineated by the services they perform and the resources they control during service sessions.

Readiness Reporting and Built-In Test. When in combat, a ship must be combat ready; this is most difficult and most necessary when the ship has just been damaged. Part of being combat ready is knowing ship status. It must be possible to determine automatically the current readiness of the ship, at any moment, to know what role the ship can play in the defense of itself and the battle force.

BMA engineers will specify *what* resources must report to support readiness assessment; resource designers will determine *how* to generate the reports. In other words, BMA will take advantage of built-in test (BIT) and built-in-test equipment (BITE), and through reporting standards, will strongly encourage the use of BIT and BITE; but the design of BIT and BITE is done by resource designers. Given the critical nature of readiness assessment in resource allocation and subsequent correct operations, BIT and BITE can be considered enabling technologies.

Example BMA Application. Figure 10 represents a conceptual application of BMA to a surface combatant. In this example, the major shipboard spaces all contain a local area network, the bridge LAN, CIC/CDC LAN, processor LAN, and intel LAN, which are all tied together through gateways by a shipboard LAN. The radio room and the engine room, as well as other shipboard spaces, are linked directly to the shipboard LAN. All major shipboard (and remote) sensors and weapons are interfaced to a high-speed data transfer network and are switched via a high-speed crossbar switch. This data transfer network is controlled via the CIC/CDC LAN, and data are entered into the shipboard processing systems via a bank of high-speed I/O processors. This sensor/weapon arrangement allows complete flexibility in warfare configuration and because of a standard interface to the data transfer network, sensors and weapons can be developed independently of the other shipboard systems and "plugged" into the architecture as they are available.

The family of LANs on the ship allows commander/user access to any other system, sensor, weapon, or processor aboard the ship. Not only is this accessibility attractive, but the systems' computers can be physically located virtually anywhere on the ship rather than be forced to reside, for example, in the CIC/CDC or Computer Room. This will increase the survivability of critical functions and systems.

The human computer interaction (HCI) associated with this architecture is intended to be such that any operator will not only have access to all shipboard systems, but will also be capable of operating any system and controlling any shipboard function.

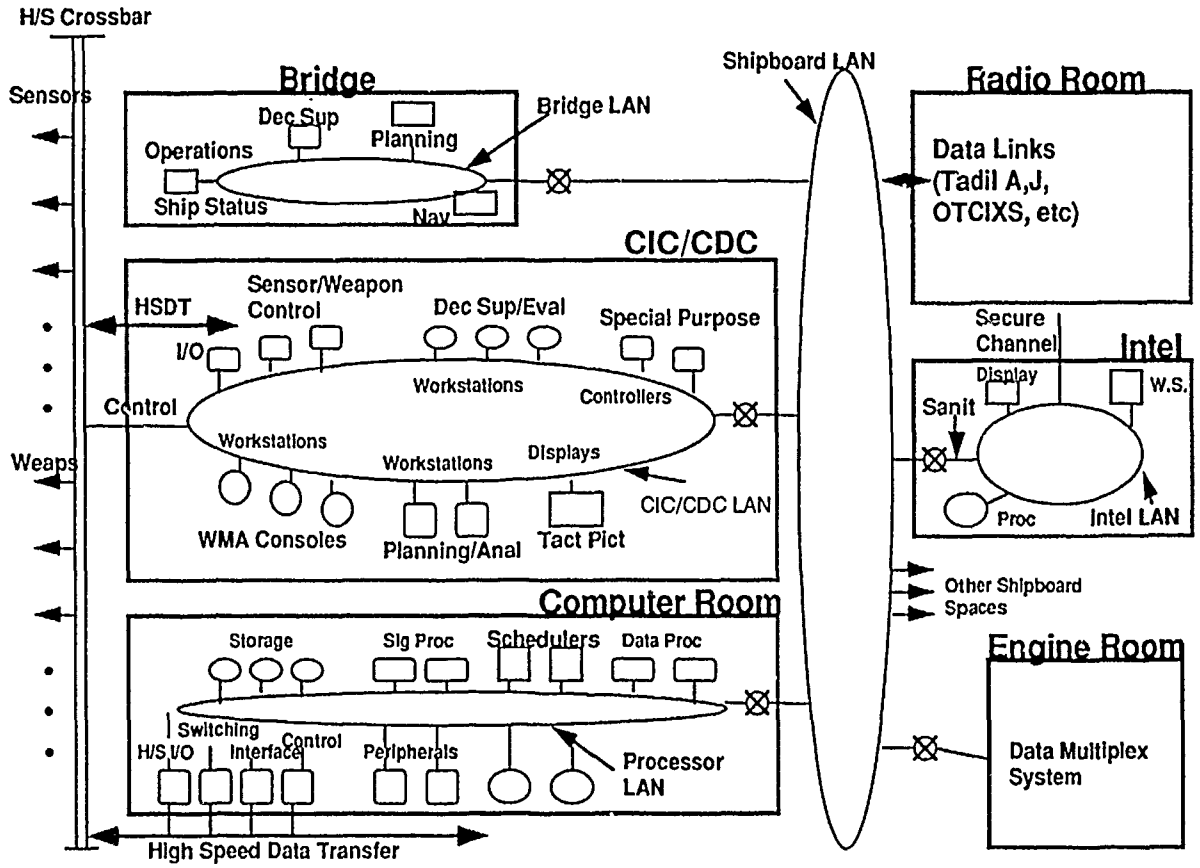


Figure 10. Example of BMA concepts applied to a surface combatant architecture.

Note that the CIC/CDC is not partitioned into warfare areas. This is because the workstations and consoles are interchangeable and allow any warfare commander or operator to conduct any warfare task. It is assumed that defense, offense, and all warfare mission area tasks would be conducted simultaneously, whether in peacetime, in wartime, or in a transition period, and therefore the architecture has been defined to maximize this flexibility.

There is much standardization implied in the figure. The software and hardware interfaces are standard. HCI is predominantly standard from system to system. The hardware, such as the workstations and processors, are nc defined to be standard, but the executive functions within those systems are standard (such as operating systems, protocols, and drivers). For instance, though one might find workstations from different manufacturers side by side in the architecture, they will possess total interchangeability and interoperability.

GENERAL CHARACTERISTICS OF THE ARCHITECTURE AND SUPPORTING ATTRIBUTES

The BMA has been described in the previous sections by the concepts utilized in its definition. In this section we desc 'be BMA by its characteristics and the attributes which support those characterist. The BMA characteristics are the generalized descripters of the architecture which can be used as design guidance or design goals. These BMA characteristics are general and apply to ashore and afloat architectures equally. The attributes are intended to support the BMA characteristics by defining the required qualities of the systems themselves.

Characteristics of the Architecture

The general characteristics and design goals of BMA are listed below:

- A layered design
  - Layers can be individually developed
  - High degree of modularity
- Service request driven
  - Well-defined interface between user/operator and physical architecture
  - Comms, information transfer and storage are "virtual"
  - User/operator need not know how architecture is designed
  - User/operator is integrated into the architecture
- Processing/information exchange are distributed
  - Processing load can be dynamically redistributed
  - Vital functions are protected
- Flexible
  - Ultimately the entire architecture is accessible to the user
  - Systems such as displays, workstations, processors, memory modules, etc. are interchangeable
  - Standard formats, protocols, interfaces and links allow internal and external (joint/allied) compatibility
  - Interoperability of information exchange is a requirement for all systems in the architecture

- Open system design
  - The architecture is expandable to adopt new technologies
  - The architecture supports simplified modification and the evolution of systems
  - The architecture is adaptable to changes in functional requirements
- Self-monitoring
  - BIT (built-in testing) is incorporated into the architecture
  - The architecture is capable of self-damage assessment
  - Resources can be reconfigured to retain critical functionality
- 3-D volumetric command displays
- Virtual reality capability
  - User training is achieved through simulation capabilities of the architecture
  - Event reconstruction and analysis is an integral part of the architecture
  - Tactical analysis and planning support are available to all commanders/ users

BMA Attributes

The BMA attributes are as follows:

1. Flexibility
   - Support all levels of conflict
   - Support transition between levels of conflict
   - Support multiple command structures
   - Support skip-echelon management
   - Support dynamic reallocation of command responsibilities
   - Support portability of command activity to alternative locations

2. Survivability
    - Support degraded modes of operation
    - Provide capability to rapidly reconstitute essential capabilities
    - Maintain continuity of information
3. Connectivity
    - Support command selectable access to needed information/data
    - Support transmission, receipt and distribution of orders, replies, information, requests for direction and requests for information
4. Interoperability
    - Support naval allied, joint, and combined operations
    - Use common information definitions/structures
    - Support effective information exchange among all users
    - Support electromagnetic compatibility
5. Integrity
    - Maintain command accountability
    - Support traceability
    - Support measurability
6. Capacity
    - Support operations under peak demand
    - Support operations throughout each mission
7. Responsiveness
    - Deliver information when needed
    - Provide timely access to high-priority information
8. Security
    - Support multiple security levels
9. Affordability
    - Provide modularity/common interfaces
    - Provide easily maintained design structures

10. Adaptability

- Support evolutionary change
- Support multiple tasking

11. Supportability

- Provide for status and effectiveness testing
- Provide on-line documentation, training and exercise necessary for maintenance
- Provide capability to defer maintenance without loss of operational capability

12. Availability

- Provide capabilities which are ready to use when needed
- Support three-level maintenance.

ENABLING TECHNOLOGIES

CONCEPT

Enabling technologies represent more than new and improved widgets. They must be examined from the perspective of the user, who is faced with the burdensome responsibility of making prudent decisions in the "fog of war." This is a time-constrained process; the user has only a limited amount of time to receive information, assess the situation, formulate a plan, make a decision, and communicate that decision to execution forces. Once this process is complete, the execution forces must have sufficient time to carry out the plan.

The problem may be viewed as a time equation. The *time available* side of the equation, which sets the problem time constraints, reflects and is influenced by threat capabilities and actions. The *time required* side of the equation, reflecting what friendly forces are capable of, includes communication time, decision time, and response time. The equation is very seldom in balance (*time available* = *time required*). When *time required* > *time available*, friendly forces are at a tactical disadvantage; otherwise they have a tactical advantage.

Emerging threat capabilities have significantly reduced the *time available* so that the equation starts out with a significant imbalance in favor of the threat. One role of enabling technology is to reduce the *time required* side of the equation and to regain the

tactical advantage. A realistic assessment of the effect of enabling technology or emergent technology upon a system must include an assessment of the effect upon this time equation.

SOME PRINCIPAL TECHNOLOGIES, HCI, AND TECHNOLOGICAL REQUIREMENTS TO SUPPORT THE ARCHITECTURE

In this section are listed some of the technological areas which have been identified as primary areas of focus to enable a transition to a BMA.

SOFTWARE ENGINEERING TECHNOLOGIES

- Software engineering tools to facilitate software design of low-cost, maintainable software.
- CASE tools to support design requirements/design structure in accordance with standards.
- User interface design tools in accordance with interface standards.
- Software testing tools to perform compatibility checks before integration.
- Computer security techniques to support multilevel data storage and access.
- Modular software development using object-oriented programming techniques to better formulate a model of reality, and to promote software reusability.

(NOTE: The major challenge and enabler in software development is automated code generation, which requires AI and other sophisticated software development techniques.)

PARALLEL COMPUTING TECHNOLOGIES

- Massive parallel (very high power) computing.
- Limited parallel supercomputing (not software limited).

(NOTE: Exploiting parallel computing architectures is a very promising approach to achieving major gains in computing power and reducing cost. Parallel computing concentrates the power of many small processors tightly coupled together. Some approaches are MIMD, SIMD, VLIW, systolic, and specialized parallel.)

NETWORK TECHNOLOGIES

- Realtime access to data for processing by all interconnected systems.
- Rapid diffusion of information among network nodes.
- System interconnectivity among nodes of differing characteristics.
- Combined voice and data applications within networks.
- Extremely wide bandwidths for information transfer channels.
- High-data-rate image transmission in addition to voice and data applications within the network.

DISPLAY TECHNOLOGIES

- High-definition displays, including HDTV.
- Real time display processing to enhance display refresh rate.
- Data compression of display information to increase display data flow rates.
- Image processing rather than display information processing as a means of increasing display refresh rates.
- Automated graphics generation for network transmission.
- Visual systems, including image generation and display which depend on improvements in field-of-view capabilities and realistic detail, including 3-D.
- Application of virtual reality display technology, a 3-D environment which can be explored, changed, experienced, and shared with others.

DECISION SUPPORT TECHNOLOGIES

- High-speed models and data processing techniques, including video disc storage of precomputed information, to result in a decrease in time required to perform complex calculations.
- Virtual simulation and modeling, to support alternative formulation and selection.
- Computer-aided design and engineering of decision support tools to optimize timeliness.
- Data association and fusion techniques to support the human decision process.

- Expert systems technology, including pattern recognition and identification.
- Intelligent decision aids that "learn" from previous experiences and are flexible enough to adjust the tactical situation and changing rules of engagement.

DATABASE MANAGEMENT/CONTROL TECHNOLOGIES

- Data "hiding" techniques to reduce the potentially disastrous effects of increased information volume and complexity.
- Distributed database techniques to avoid duplication while maintaining flexibility and to avoid storing inconsistent information.

OPERATOR/MACHINE INTERFACE TECHNOLOGIES

- Integrating personnel with high technical skills into the architecture to increase the knowledge base.
- Advances to enhance human computer interface to more accurately simulate natural experience.
- Natural language understanding (NLU).
- Capabilities to allow human to instinctively interact with the machine.
- 3-D volumetric displays that approximate reality.
- Embedded multilingual channels for communications among speakers of different languages (potentially embedded in displays).
- Instructor station capabilities, along with simplification of instructor's task to allow more time for management of simulation exercises and applications.

STANDARDS ESSENTIAL TO THE DEVELOPMENT OF BMA

Standards, generally in the form of hardware and software interfaces, formats, operating systems, and software development tools are central to developing a modular, open, and cost-effective architecture. Standards created too early in a technology development, however, could inhibit future advances and "lock-in" less than optimal solutions. On the other hand, an absence of standards could cause delayed access to information, longer development time, higher cost of development, and poor coordination of architectural development. It is also crucial in the development of standards to provide a means for their incremental upgrade and logistical support.

Some standards currently under development which are pertinent to an implementation of BMA are those of the Next Generation Computer Resources (NGCR) program, which is under the direction of the Space and Electronics Warfare Systems Command.

Components being developed by NGCR are—

- Multiprocessor Interconnects
  - Backplane
  - High-speed data transfer network
  - High-performance backplane
- Multisystem Interconnects
  - Local area network
  - High-performance local area network
- Software Standardization Areas
- Operating System Interface
- Database Management System Interface
- Programming Support Environment
- Graphics Language/Interface

In addition to the components of NGCR, there are other general areas which have been recommended to be standardized to support the architectural transition to BMA, namely—

- Hardware/software interfaces, tieing developmental C^2 systems with existing combat systems.
- Interchangeable hardware/software building blocks to promote the portability and replaceability of hardware or software as technology warrants.
- Protocols/data packaging/messages/information exchange formats to promote increased data flow rates among and within nodes.
- Data and control busses to manage the flow of information among and within nodes.
- HCI, readiness and status reporting formats and procedures that reflect operational and tactical requirements.
- Data storage techniques that optimize performance while promoting consistency among diffused databases.
- Data switching that might result in increased flow rates from high-volume or high-rate input sources.

DEVELOPMENT SUPPORT

One step toward realizing the objectives of this effort is to establish one or more BMA Node Test Sites. These Test Sites will be the product of system engineering, begun in the BMA design process. A key part of the system engineer's job will be to enforce BMA standards. The sites may be developed from existing facilities but must be governed by BMA guidelines and be capable of accommodating the entire BMA product line.

Once the test sites are operational, it will be possible both to mandate that BMA products conform to BMA standards and to verify that they do. These test sites can become key stepping stones to the Fleet for a fully integrated command and control communications and intelligence (C^3I) product line. Figure 11 illustrates this principle.

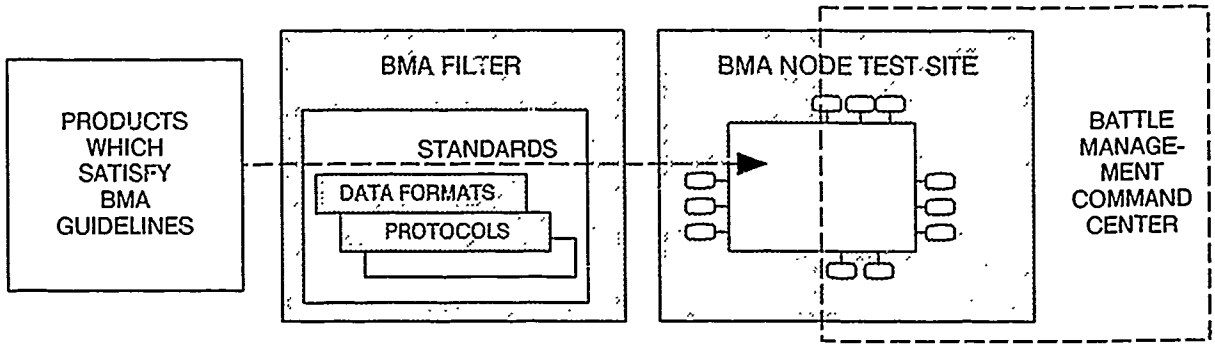


Figure 11. BMA test node site.

The BMA Node Test Site performs integration testing, analysis, and the evaluation of the architecture. The Battle Management Command Center is a mock-up that includes interfaces from command and control decision support systems to physical weapons and sensing systems, and the stabilized prototype layers of BMA, existing at the Node Test Site (which represents a baseline). The Command Center mock-up performs two specific functions: (1) it stimulates the BMA Node Test Site as a higher, lower, or equivalent echelon command; and (2) it provides a vehicle for the assessment of emergent technology at all BMA architectural layers. The baseline systems run in a side-by-side configuration with the Command Center mock-up.

REFERENCES

1. Wessel, P. et. al., *Battle Management Architecture 2015, 'A way forward,' Volumes 1 and 2*, SPAWAR 30, October 1989.

2. Simon, H. A. (1962). The Architecture of Complexity. *Proceedings of the American Philosophical Society, 26*, 467-482.

3. Brooks, R. A. (1986). A Robust Layered Control System For A Mobile Robot. *IEEE Journal of Robotics and Automation, RA-2*(1), 14-23.

4. CSS Program Office, NOSC Code 8503, *Communication Support System (CSS) Overview*, Naval Ocean Systems Center, 26 July 1990.

5. Military Handbook (MIL-HDBK-0036, Draft): "Survivable Adaptable Fiber Optic Embedded Network II, SAFENET II."

Appendix A

DEFINITIONS

Allocation—(see *Resource allocation*).

Application—a process which directly supports one or more services.

Architecture—a set of functions, the processes and resources which perform those functions, and the interrelationships among those functions, processes, and resources.

Attribute—technical features that provide the underlying framework for BMA.

Backplane—a panel, generally located at the rear of an integrated-circuit-card enclosure, housing one or more (backplane) buses which support communication among the cards in the enclosure.

Battle Management Architecture (BMA)—a top-level physical, functional, and organizational structure which provides the abilities to interact with commands and their assets to effectively employ and control U.S. Navy, joint, and combined forces. It ties together the systems for directing and controlling naval forces in peacetime, in preparation for and during battle, and it includes all levels of command from the NCA to the individual fighting element. It will include and interrelate sensors, intelligence, command and control, combat direction, weapon system control, electronics warfare, and supporting logistic elements.

C^2 interface—see *Command and control interface.*

C^3I—see *Command and control communications and intelligence.*

Central Processing Unit (CPU)—that part of a computer which performs data transformation.

CIC/CDC—the Combat Information Center (CIC) is now called the Command Direction Center (CDC).

Command—an order, passing from the functions layer (the user) to the services layer or from one decision support element to a second decision support element at a lower level node, for which the receiving decision support element is capable of supplying a corresponding system service and returning a response.

Command and control communications and intelligence—the fusion of intelligence, surveillance, and tactical information into a common and consistent tactical picture.

Command and control interface (or C^2 interface)—an interface which enables a user to (1) enter commands, by translating functions into command sequences or by

formatting individual commands, (2) assign priorities to commands, (3) partition command repertoires into subsets, (4) delegate command subsets to subordinate users, and (5) delegate user-priority-range subsets to subordinate users.

Command node—see *Node*.

Communication element—the element which exchanges information with other nodes.

Communication Support System (CSS)—a SPAWAR PD-50 sponsored program, intended to define a Navy communications subarchitecture.

Cooperative engagement—a warfighting capability designed to defeat threats through the synergistic integration of distributed resources among two or more units.

CPU—see *Central Processing Unit*.

CSS—see *Communication Support System*.

Database management—the storage, update, and retrieval of related sets of data, generally accomplished by a database management system (DBMS).

Deadlock—a condition where, in a set of two or more services, each is halted, waiting for one or more resources controlled by one or more other members of the set.

Decision support element—the element which supports command planning and decision making through assessment and evaluation and which accepts commands, provides system services, and returns responses.

Element (or Resource element)—an entity defined with respect to a set of services it provides, and which incorporates an element controller.

Element controller—an application within an element, active at all times, that handles interelement communication, service scheduling, and resource allocation and control during service sessions.

Element service—the mechanism by which support is supplied by an element to the decision support element in response to a request.

Enabling technology—a technology, identified subjectively, which allows critical features of a system to be implemented.

Engagement element—the element which destroys or neutralizes external threats.

File-server—a device which stores and retrieves files, and transmits them on request, generally over a LAN, to a requesting device, such as a workstation.

Finite-state machine—a mathematical model of a system which (1) has a finite number of internal states, (2) may accept one of a finite number of inputs, (3) enters a new

state—or remains in the current state—based on the current state, the input, and a state transition function, and (4) produces an output according to the current state, the input, and an output function.

Function—that which a user does.

General-purpose resource—a resource which supports a broad range of applications and utilities.

High-speed switch—a general-purpose resource capable of directly connecting two resources, e.g., a sensor or weapon and a processor.

LAN interface—the hardware and software which allow a device to connect to and make use of a local area network.

Layered architecture—an architecture, partitioned "vertically" into layers so that each layer supports the next higher layer.

Local area network (LAN)—a general-purpose resource, including one or more cables and the associated interfaces (hardware and software) that enables devices which observe interface specifications to connect (to the LAN) and to communicate (via the LAN).

Mass storage device—a secondary storage device, capable of storing very large amounts of data.

Memory—that part of a computer which holds instructions and data directly accessible by the CPU.

Metropolitan area network (MAN)—a backbone network, interconnecting local area networks.

Monitor—that part of a computer which includes a display screen and which is generally controlled by and allows interaction through a keyboard, a mouse, a touch panel, etc.

Navigation element—the element which plans, records, and controls node position and velocity.

Network partition—when one or more nodes on a network cannot communicate with all other nodes on the network.

Next Generation Computer Resources (NGCR) Program—a SPAWAR 324 Program, the purpose of which is to define utility and general-purpose hardware standards for the Navy.

Node (or *Command node*)—(1) embodies all the physical aspects of the command, including the commander, related to battle management, (2) is the physical construct in which the command process is performed, and (3) is decomposed into elements which may be distributed.

Node ID—an identifier, composed of theater, force, and platform identification fields, one or more of which may be null, depending on the level of the node.

Open System—a layered architecture the interfaces of which are defined by industry standards, controlled by recognized standards organizations (ISO, ANSI, IEEE, SAE, etc.).

Operating System—general-purpose software which makes the hardware resources usable by the utilities and applications.

Preemptable—characteristic of a resource which can be preempted because the priority of a new service exceeds the current resource priority.

Preemption—the withdrawal of resource access from one element by a second element to support a higher priority service.

Priority—a characteristic of a command, supplied by the user, inherited by all supporting requests and services, and used to resolve contention for resources.

Priority Range—a set of priorities, assigned to a user, which determines what priorities that user may attach to commands.

Process—software, capable of executing on one or more resources, designed to support one or more functions.

Replay—information (acknowledgment, "cannot comply," etc.) returned by an element controller in response to a request.

Request—an order for service placed on an element by the decision support element, for which the performing element is capable of supplying a corresponding service and returning a reply.

Resource—hardware, either special-purpose (sensor, weapon, etc.) or general-purpose (local area network, workstation, etc.), necessary to support processes.

Resource allocation—the granting of resource access.

Resource element—(see *Element*).

Response—information (acknowledgment, "cannot comply," etc.) returned from the services layer (the decision support element) to the functions layer in response to a command.

Segmentation (of commands)—a process of dividing commands into subsets sufficient to support distinct areas of responsibility.

Sensing element—the element which detects and characterizes objects and characterizes the environment external to the node.

Service—a system service or an element service.

Service session—the period during which an element performs a service and retains control of resources necessary to perform that service.

Signal processing—transformation of an incoming signal, by means of numeric processing, to extract information; signal processing tends to be computation intensive.

Signal processor—a device designed to perform (computation-intensive) signal processing.

Special-purpose resource—a resource which directly supports particular element services.

Subordinate user—a person authorized by a user at a node to access the BMA via the C2 interface and provided by that user with a command repertoire and a priority range.

Subsumption architecture—a functional decomposition by task-achieving system behaviors, where each task-achieving behavior is implemented explicitly as a stable (subassembly) level in the hierarchy. Each level represents a level of competence (with respect to performance of specific tasks at that level) which enables the construction of a control system with layers corresponding to each level of competence.

System service—the mechanism by which support is supplied by the services layer (decision support element) to the functions layer in response to a command.

System-service-based resource allocation policy—a scheme to prevent deadlock wherein a system service cannot begin until complete resource control has been established by all elements with respect to all element services necessary to support the system service.

Utility—a process which supports a broad range of applications.

User—a commanding officer of a node or a subordinate authorized by the commanding officer to interact with the BMA via the C2 interface.

Workstation—a device which includes a backplane, a memory, a CPU, a monitor, and a LAN interface and is used primarily for data processing.

Appendix B

THREATS

THREATS TO NAVAL FORCES

- Surface and/or subsurface targets beyond the horizon
- Air targets at low altitudes
- Air, surface, and/or subsurface targets with significantly reduced signature characteristics
- Platforms and weapons approaching at supersonic speeds
- High-altitude targets (above weapon capability)
- Threat conditions requiring emission control
- High raid-count air threat
- Simultaneous multimedia raid (up, out, and down)

A list of architectural features, including areas that an evolving architecture must support, is provided below:

- Multiple Command Levels—Global, Theater, Force, and Platform
- Multimedia—Space, air, sea surface, underwater, and land
- Multispectral—Covers all bands in the frequency spectrum (FLIR, Laser, Radar, Sonar, ESM, etc.)
- Multiwarfare—AAW, ASW, ASUW, SEW, etc.
- Independent, Coordinated, and Cooperative Operations
- U.S. Navy, Joint, and Combined Operations

Another way of presenting the situation is to divide things into "problem threat (platforms and weapons) characteristics" and "problem U.S. Navy response characteristics," as shown below:

PROBLEM THREAT CHARACTERISTICS

Target Kinetics

- Approaching at low elevation
- Approaching at high elevation
- Approaching at supersonic speed
- Capable of high acceleration (maneuverability)

Stealth

- Target constructed to have reduced signature characteristics
- Targets attack with advantage of environmental and/or ECM interference

PROBLEM U.S. NAVY RESPONSE CHARACTERISTICS

Need to preserve covertness (emission control)

Surveillance (spectrum coverage)

Weapon prioritization

The things we are trying to achieve are—

- Increased number of engagement opportunities
- Increased engagement ranges
- Preservation of U.S. Navy platform covertness

This should result in both extending the battle space and increasing firepower.

Appendix C

EXAMPLES OF LAYERING

IBM SYSTEM/360

In the early 1960s, the International Business Machines Corporation (IBM) embarked on a strategy to meet the needs of customers, which revolved around a new computer architecture, the System/360. The strategy was so successful that, to this day, the IBM System/360 architecture remains the basis for billions of dollars of business and the heart of automatic data processing (ADP) for thousands of corporations and companies around the world.

The key to the System/360 is the (remarkably foresighted) concept of layering. At the lowest layer, hardware, IBM placed their S/360 Instruction Set Architecture (ISA). While the hardware evolved from the S/360 Model 40 through Models 65, 75, 85 (with cache memory), and 91 (with pipelined CPU), into the S/370 series, and beyond, each model supported the instruction set of previous models (i.e., was upward compatible).

At the second layer, OS/360 was built directly on the S/360 ISA. This allowed OS/360 to provide a stable set of services to higher levels even as new hardware models were introduced.

At the third layer are the utilities (language processors, database management programs, etc.) necessary for a variety of customer applications. These rely on a stable set of OS/360 services as well as on the S/360 ISA.

The fourth layer, applications, represents for many customers a very large capital investment, made over a period of years. As more and faster service is required, customers can upgrade the ADP system at any layer with minimal change to applications.

This short discussion is a case for layering, not for the IBM S/360 per se. Layering has allowed IBM customers to upgrade the level of ADP service while protecting them from destructive change to their applications. And, it has allowed IBM to retain a "captive" customer base. But, time has revealed a missing dimension for customers—*open systems*, which enhances layering by expanding the supplier base, the key to avoiding captivity.

OSI REFERENCE MODEL

The International Standards Organization (ISO) proposed an Open Systems Interconnection (OSI) Reference Model. This model, depicted in Fig. C-1, is intended

to support the connection of "open systems," those which are open for communication with other systems. The following principles apply: a layer was created wherever a different level of abstraction was needed; each layer performed a well-defined function; layer functions were chosen in anticipation of internationally standardized protocols; layer boundaries minimized information flow across interfaces; the number of layers was large enough to separate distinct functions into separate layers and small enough to keep the architecture manageable.

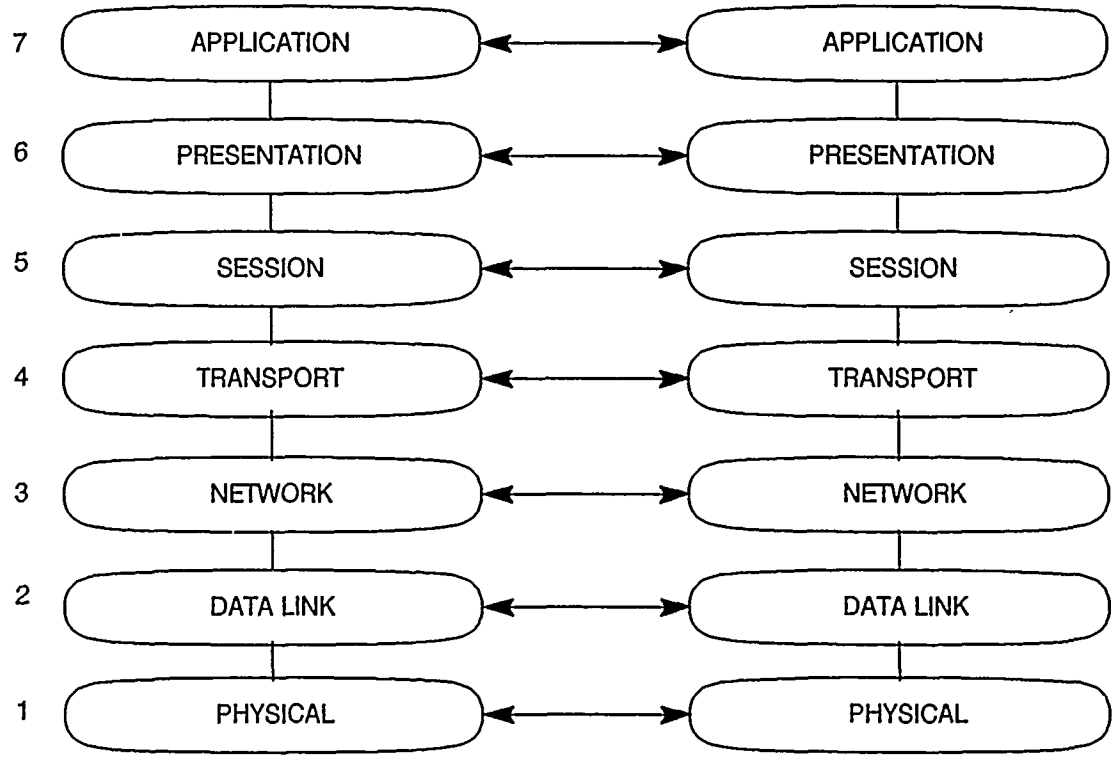


Figure C-1. OSI seven-layer reference model.

An excellent reference on the OSI Reference Model is *Computer Networks* by Andrew S. Tanenbaum (Prentice Hall, 2nd Edition, 1988).

The *Physical Layer* is concerned with transmitting bits over a channel. Voltage levels for binary 0 and binary 1 are defined. Cycle times and bit rates are defined. Mechanical, electrical, and procedural interfaces—connector configurations, including numbers of pins and what each pin is used for, electrical currents, etc.—are designed to be compatible with bit transmission rates and directions. The physical layer is generally considered the domain of the electrical engineer.

The *Data Link Layer* provides error-free data transfers in support of the network layer. Outgoing data may be divided into data frames (several hundred bytes long); in

some networks, an acknowledgment is expected from the receiver. A second key function in the Data Link Layer is network access. Various schemes, such as time-division multiple access (TDMA), Carrier Sense Multiple Access (CSMA), and Token Passing, are used to share access to the channel among network nodes.

The *Network Layer* sets up routes in a network to move packets from sender to receiver. In networks on a single bus, this is a trivial function. In radio networks, where relays are often required, this can be extremely complex. The complexity also varies according to whether packets are intended for a single receiver (point-to-point), several receivers (multicast), or all nodes in a network (broadcast). A related function at the Network Layer is flow control, assuring that packets are routed to avoid bottlenecks and minimize queueing delays wherever possible.

The *Transport Layer* assures the reliable transfer of data between users. In other words, the Transport Layer is considered an "end-to-end" layer. Outgoing data are packetized. Incoming packets are reassembled in correct order; discrepancies are noted and may be reconciled with the sender (through acknowledgments, negative acknowledgments, or requests to retransmit). Error detection and correction codes may be used to minimize bit errors and resulting retransmissions.

The *Session Layer* allows users at different nodes to establish virtual circuits, or "sessions," between them. A user might log onto a remote time-shared computer or establish a file transfer. Two or more users might hold a conversation or conduct a dialogue at data terminals. Mechanisms to establish, maintain, log, and disestablish a session are incorporated into this layer.

The *Presentation Layer* performs certain functions that support many users. The encoding of character strings (e.g., into ASCII or EBCDIC) and the support for other standard presentation formats are incorporated into this layer. Data compression would be supported at the Presentation Layer. This layer should not be needed, since BMA is expected to use common message and data formats.

The *Application Layer* handles various (otherwise incompatible) user terminals and supports file transfer, electronic mail, remote job entry, and a variety of other functions required by the user, whether or not this node is on a network. The Application Layer must determine whether required resources are local (at this node) or remote (at another node on the network).

Appendix D

FUNCTIONS AND SUBFUNCTIONS

The following sets of functions, at the force and platform levels, are presented as an example. A more refined set, capable of standing as a candidate for adoption under the subsumption architecture, is under development. For this example, the commander performs the five high-level functions—decision support, sense, engage, navigate, and communicate—at each level. These are decomposed at the various levels into subfunctions.

Global Level

To be determined

Theater Level

To be determined

Force Level

Communicate

- Send message
- Establish virtual circuit

Assess and Report Readiness

- Assess and report force operational capability
- Assess and report force maintenance requirements
- Assess and report force inventory status

Navigate

- Determine current force position and velocity
- Transit force
- Establish positions for point and area defense

Sense

- Maintain force surveillance picture
- Allocate surveillance sectors to platforms
- Maintain environmental data

Detect, Track, and Discriminate

Collect platform track reports
Identify threats
Classify targets
Determine raid size
Evaluate threats

Develop and Maintain Tactical Picture

Correlate, associate, and fuse tracks
Present force tactical picture

Command and Control

Control platform positions
Control platform emissions

Engage Targets

Select tactical response (hard or soft kill)
Assign platforms/weapons to targets
Order independent, coordinated, or cooperative engagement

Assess Battle Damage

Assess damage to hostile forces
Assess damage to own force

Support Decisions

Report readiness of all or selected platforms
Report current force position and velocity
Display all or selected contacts
Provide priority tnreat ranking
Provide priority platform ranking

Platform Level

Assess and Report Readiness

Assess and report platform operational capability
Assess and report platform maintenance requirements
Assess and report platform inventory status

Navigate

Determine current position and velocity
Vector and maneuver

Sense

"Look" (for space, air, and surface targets)
"Listen" (for subsurface targets)
Collect environmental data

Detect, Track, and Discriminate

Detect contacts
Identify threats
Classify targets
Determine raid size
Evaluate threats (before weapon assignment)

Develop and Maintain Track Picture

Associate data
Correlate data
Fuse data

Command and Control

Control sensors
Control weapons
Reposition platform

Engage Targets

Hard Kill
- Prioritize weapons
- Assign weapons (after threat evaluation)
- Launch, fire weapons
- Control weapons as required

Soft Kill
- Employ electronic countermeasures (ECM)
- Change platform position, aspect, or velocity

Assess Battle Damage

Enemy (kill assessment)
Own force
Own platform

Support Decisions

Report readiness of all or selected resources
Report current position and velocity

Display all or selected contacts
Provide priority threat ranking
Provide priority weapon ranking

Appendix E

COMMANDS, REQUESTS, AND SERVICES

This is an example set of commands, requests and services to implement the function set in Appendix D at the force and platform levels. No attempt has been made to aggregate commands; aggregation is expected to take place at the functions layer.

Each command, request, or service carries the designation XYZnn, where

X = {R, S}, request, command or service
Y = {P, F, T, G}, the initial of the Level
Z = {D, S, E, N, C}, the initial of the serving element
nn = a decimal identification number

Each service is assigned an internal identifier, which is returned to the requester in an acknowledgment ("ack"). The identifiers include the requesting element name and level and the command priority. Identifiers are used to monitor or cancel services. In the listings below, services are related to supporting requests.

In the discussions below, subfunctions are shown in italics. A command carries the identifier of the element which provides the service. This appendix is divided into 6 parts. The first part maps functions into commands. The next five parts show the services listed by performing element and the requests which the performing element must make in order to provide the service.

MAPPING FUNCTIONS INTO COMMANDS

Force Level

Select tactical force response (hard or soft kill)

RFD02—Report track status (in: qualifiers, track; out: track)

RFD05—Recommend tactical force response (in: track; out: kill type)

Assign platforms/weapons to targets

RFE01—Assign platforms/weapons to target (in: track, kill type; out: platforms/weapons, engagement type)

Conduct independent engagement

RFE02—Initiate independent engagement (in: track, platforms, kill type; out: ack)

RFE04—Terminate independent or coordinated engagement (in: track, platforms; out: ack)

Conduct coordinated engagement

RFE03—Initiate coordinated engagement (in: track, platforms, kill type; out: ack)

RFE04—Terminate independent or coordinated engagement (in: track, platforms; out: ack)

Conduct cooperative engagement

RFE05—Initiate cooperative engagement (in: track, platforms, kill type; out: ack)

RFE06—Terminate cooperative engagement (in: track, platforms; out: ack)

Change force position, aspect, or velocity

RFD01—Report force position (out: platforms)

RFD07—Determine force aspect (in: screen/search reqs, etc.; out: relative platform positions)

RFN01—Establish force velocity (in: speed, bearing; out: ack)

RFN02—Move force to a fixed position (in: coordinates; out: ack)

RFN03—Maintain contact with a track (in: track; out: ack)

RFN04—Maintain force position relative to a track (in: track; out: ack)

Manage force emissions

RFD10—Report force emissions/comm plan (in: force emission constraints; out: force plan)

Maintain force position picture

RFD01—Report force position (out: platforms)

Transit force

RFN01—Establish force velocity (in: speed, bearing; out: ack)

RFN02—Move force to a fixed position (in: coordinates; out: ack)

Establish positions for point and area defense

RFD07—Determine force aspect (in: screen/search reqs, etc.; out: relative platform positions)

Maintain force surveillance picture

RFD08—Determine force surveillance requirements (in: volume; out: platforms/volumes)

Allocate surveillance sectors to platforms

RFS01—Assign volume to platform (in: platform, volume; out: ack)

Detect potential targets

RFS01—Surveil a volume (in: volume; out: ack, track)

RFS02—Monitor track, search quality (in: track; out: ack, track)

RFS05—Identify track (in: track; out: track (with ID))

RFS06—Assign volume to platform (in: platform, volume; out: ack)

Localize, classify, and identify track

RFD03—Integrate track (in: track; out: track)

Associate, correlate, and fuse multiple track data

RFD03—Integrate track (in: track; out: track)

Present force tactical picture

RFD01—Report force position (out: platforms)

RFD02—Report track status (in: qualifiers, track; out: track)

RFD03—Integrate track (in: track; out: track)

Evaluate threat to force

RFD04—Rank threats to force (out: platforms/track)

Maintain Environmental Data

RFD09—Report environment (out: report)

Assess damage to hostile forces

RFD02—Report track status (in: track; out: track)

Assess damage to own force

RFD20—Report force status (in: type: out: report) type = {*operational, maintenance, inventory*}

Assess and report force operational capability

RFD20—Report force status (in: *operational*: out: report)

Assess and report force maintenance requirements

RFD20—Report force status (in: *maintenance*: out: report)

Assess and report inventory status

RFD20—Report force status (in: *inventory*: out: report)

General services which support the user directly

Voice Communications

RFC02—Establish virtual circuit (in: "voice," caller address, callee address; out: circuit designation)

RFC03—Disestablish virtual circuit (in: circuit designation)

Mail Processing and Management

RFC05—Send electronic mail message (use SFC01 format)

RFC06—Read electronic mail messages (in: user ID; out: messages)

Platform Level

Select tactical platform response (hard or soft kill)

RPD05—Recommend tactical platform response (in: target; out: kill type)

Assign (hard or soft) weapons on platform to targets

RPE01—Assign weapons (in: target, track, kill type; out: weapons)

RPE09—Release weapon control (in: weapon; out: ack)

Assign (hard or soft) weapons on platform to sectors (close-in-defense)

RPE01—Assign weapons (in: sector, coordinates, kill type; out: weapons)

RPE09—Release weapon control (in: weapon; out: ack)

Launch, fire weapons

RPE07—Launch weapon (in: target, weapon; out: ack)

Control weapons as required

RPE08—Control weapons (in: target, weapon; out: ack)

RPE09—Release weapon control (in: weapon; out: ack)

Change platform position, aspect, or velocity

RPN01—Establish platform velocity (in: speed, bearing; out: ack)

RPN02—Move platform to a fixed position (in: coordinates; out: ack)

RPN03—Move platform to a track (in: track; out: ack)

RPN04—Maintain platform position relative to a track (in: track; out: ack)

RPN05—Avoid objects (out: ack)

Manage platform emissions

RPD10—Report platform emissions/comm plan (in: platform emission constraints; out: platform plan)

Navigate

RPN01—Establish platform velocity (in: speed, bearing; out: ack)

RPN02—Move platform to a fixed position (in: coordinates; out: ack)

RPN05—Avoid objects (out: ack)

Vector and maneuver

RPN03—Move platform to a track (in: track; out: ack)

RPN04—Maintain platform position relative to a track (in: track; out: ack)

RPN05—Avoid objects (out: ack)

Maintain platform surveillance picture

RPS01—Surveil a volume (in: volume; out: ack, track)

Detect potential targets

RPS01—Surveil a volume (in: volume; out: ack, track)

RPS02—Monitor track, search quality (in: track; out: ack, track)

RPS05—Identify track (in: track; out: track (with ID))

Manage platform track store

RPD02—Report track (in: qualifiers, track; out: track)

Localize, classify, and identify track

RPD03—Integrate track (in: track; out: track)

Associate, correlate, and fuse multiple sensor data

RPD03—Integrate track (in: track; out: track)

Establish raid size

RPD03—Integrate track (in: track; out: track)

Evaluate threat to platform

RPD04—Rank threats to platform (out: track)

Maintain Environmental Data

RPD09—Report environment (out: report)

Assess damage to engaged targets

RPD02—Report track status (in: track; out: track)

Assess damage to own platform

RPD20—Report platform status (in: type: out: report) type = {*operational, maintenance, inventory*}

Assess and report platform operational capability

RPD20—Report platform status (in: *operational*: out: report)

Assess and report platform maintenance requirements

RPD20—Report platform status (in: *maintenance*: out: report)

Assess and report platform inventory status

RPD20—Report platform status (in: *inventory*: out: report)

General services which support the user directly

Voice Communications

OS Services

- Establish virtual circuit (in: "voice," caller address, callee addresses; out: circuit designation)
- Disestablish virtual circuit (in: circuit designation)

Mail Processing and Management

OS Services

- Send electronic mail message (use format)
- Read electronic mail messages (in: user ID; out: messages)

DECISION SUPPORT ELEMENT

Force Level

Supporting platform requests are received as commands by the platform decision support element.

SFD01—Report force position (out: platforms)

RPD01—Report platform position (out: platform)

RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

SFD02—Report track status (in: qualifiers, track; out: track)

SFD03—Integrate track (in: track; out: track)

SFD04—Rank threats to force (out: platforms/track)

SFD05—Recommend tactical force response (in: target; out: kill type)
SFD07—Determine force aspect (in: screen/search reqs, etc.; out: relative platform positions)
SFD08—Determine force surveillance requirements (in: volume; out: platforms/ volumes)
SFD09—Report environment (out: report)
SFD10—Report force emissions/comm plan (in: force emission constraints; out: force plan)
RPD10—Report platform emissions plan (in: platform emission constraints; out: platform plan)
RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)
SFD20—Report force status (in: type: out: report*)
type = {operational, maintenance, inventory}
RPD20—Report platform status (in: type; out: report*)
* Operational reports show applications available
RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

Platform Level

SPD01—Report platform position (out: platform)
RPN06—Report platform position (out: platform)
SPD02—Report track (in: qualifiers or track; out: track)
SPD03—Integrate track (in: track; out: track)
SPD04—Rank threats to platform (out: track)
SPD05—Recommend tactical platform response (in: target; out: kill type)
kill type = {hard, soft}
SPD10—Report platform emissions/comm plan (in: platform emission constraints; out: platform plan)
RPS/E/C/N10—Report/limit element emissions (in: platform emission constraints; out: plan)
SPD20—Report platform status (in: type: out: report*)
type = {operational, maintenance, inventory}
* Operational report shows applications available
RPS/E/C/N20—Report element status (in: type, resource status report; out: element status report**)
** Element operational report shows services available
* Report showing operational status of equipment

SENSING ELEMENT

Force Level

Supporting platform requests are received as commands by the platform decision support element.

SFS01—Surveil a volume (in: volume; out: ack, track)
- RFS05—Assign volume to platform (in: platform/volume; out: ack, track)
- RFD02—Report track status (in: qualifiers/track; out: track)
- RFD03—Integrate track (in: track; out: track)
- RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

SFS02—Monitor track, search quality (in: track; out: ack, track)
- RFD02—Report track status (in: qualifiers, track; out: track)
- RFD03—Integrate track (in: track; out: track)
- RPS02—Monitor track (in: track; out: ack, track)
- RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

SFS03—Monitor track, fire-control quality(in: track; out: ack, track)
- RFD02—Report track status (in: qualifiers, track; out: track)
- RFD03—Integrate track (in: track; out: track)
- RPS02—Monitor track (in: track; out: ack, track)
- RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

SFS04—Limit force emissions (in: allowable platform emissions; out: ack)
- RPS04—Limit platform emissions (in: emission type, range; out: ack)
- RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

SFS05—Identify track (in: track; out: track (with ID))
- RFD02—Report track status (in: qualifiers, track; out: track)
- RFD03—Integrate track (in: track; out: track)
- RPS05—Identify track (in: track; out: track (with ID))
- RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

SFS06—Assign volume to platform (in: platform/volume; out: ack, track)
- RFD02—Report track status (in: qualifiers, track; out: track)
- RFD03—Integrate track (in: track; out: track)
- RPS01—Surveil a volume (in: volume; out: ack, track)
- RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

SFS07—Terminate sensing service (in: identifier; out: ack)
SFS10—Report force emissions/comm plan (in: constraints; out: plan)
RPS10—Report element emissions plan (in: constraints; out: plan)

Platform Level

SPS01—Surveil a volume (in: volume; out: ack, track)
RPD02—Report track (in: qualifiers or track; out: track)
RPD03—Integrate track (in: track; out: track)
SPS02—Monitor track, search quality (in: track; out: ack, track)
RPD02—Report track (in: qualifiers or track; out: track)
RPD03—Integrate track (in: track; out: track)
SPS03—Monitor track, fire-control quality (in: track; out: ack, track)
RPD02—Report track (in: qualifiers or track; out: track)
RPD03—Integrate track (in: track; out: track)
SPS04—Limit platform emissions (in: emission type, range; out: ack)
SPS05—Identify track (in: track; out: track (with ID))
RPD02—Report track (in: qualifiers or track; out: track)
RPD03—Integrate track (in: track; out: track)
SPS07—Terminate sensing service (in: identifier; out: ack)
SPS10—Report element emissions plan (in: constraints; out: plan)
SPS20—Report element status (in: type; out: report)

ENGAGEMENT ELEMENT

Force Level

Supporting platform requests are received as commands by the platform decision support element.

SFE01—Assign platforms/weapons to target (in: track, kill type; out: platforms/weapons)
RPE01—Assign weapons (in: sector or target, kill type; out: weapons)
kill type = {hard or soft}
RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)
SFE02—Initiate independent engagement (in: track, platforms/weapons, kill type; out: ack)
RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)
RPE02—Support independent engagement (in: track, weapons, kill type; out: ack)

SFE03—Initiate coordinated engagement (in: track, platforms/weapons, kill type; out: ack))
RPE03—Support coordinated engagement (in: track, weapons, kill type, other platform; out: ack)
RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)
SFE04—Terminate independent or coordinated engagement (in: track, platforms; out: ack)
RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)
RPE04—Terminate independent or coordinated engagement (in: track; out: ack)
SFE05—Initiate cooperative engagement (in: track, platforms/weapons, kill type; out: ack))
RFC02—Establish virtual circuit (in: qualifiers, caller address, callee addresses; out: circuit)
RPE05—Support cooperative engagement (in: track, weapons, kill type, other platform, circuit; out: ack)
RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)
SFE06—Terminate cooperative engagement (in: track, platforms; out: ack)
RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)
RFC03—Disestablish virtual circuit (in: circuit)
RPE06—Terminate independent or coordinated engagement (in: circuit, track; out: ack)
SFE10—Report force emissions plan (in: constraints; out: plan)
RPE10—Report element emissions plan (in: constraints; out: plan)

Platform Level

SPE01—Assign weapons (in: sector or target, kill type; out: weapons)
kill type = {hard or soft}
RPD02—Report track (in: qualifiers or track; out: track)
SPE02—Support independent engagement (in: track, weapons, kill type; out: ack, launch report)
RPS02—Monitor track, search quality (in: track; out: ack, track)
RPS03—Monitor track, fire-control quality (in: track; out: ack, track)
RPE07—Launch weapon (in: track, weapon; out: ack)
RPE08—Control weapon (in: target, weapon; out: ack)
RPD02—Report track (in: qualifiers or track; out: track)

SPE03—Support coordinated engagement (in: track, weapons, kill type, other platform; out: ack, launch report)
RPS02—Monitor track, search quality (in: track; out: ack, track)
RPS03—Monitor track, fire-control quality (in: track; out: ack, track)
RPE07—Launch weapon (in: track, weapon; out: ack)
RPE08—Control weapon (in: target, weapon; out: ack)
RPD02—Report track (in: qualifiers or track; out: track)
SPE04—Terminate independent or coordinated engagement (in: track; out: ack)
SPE05—Support cooperative engagement (in: track, weapons, kill type, other platform, circuit; out: ack, launch report)
RPS02—Monitor track, search quality (in: track; out: ack, track)
RPS03—Monitor track, fire-control quality (in: track; out: ack, track)
RPE07—Launch weapon (in: track, weapon; out: ack)
RPE08—Control weapon (in: target, weapon; out: ack)
RPD02—Report track (in: qualifiers or track; out: track)
SPE06—Terminate cooperative engagement (in: circuit, track; out: ack)
RPD02—Report track (in: qualifiers or track; out: track)
SPE07—Launch weapon (in: track, weapon; out: ack)
SPE08—Control weapon (in: target, weapon; out: ack)
SPE09—Release weapon control (in: weapon; out: ack)
SPE10—Report element emissions plan (in: constraints; out: plan)
SPE20—Report element status (in: type; out: report)

NAVIGATION ELEMENT

Force Level

Supporting platform requests are received as commands by the platform decision support element.

SFN01—Establish force velocity (in: speed, bearing; out: ack)
RPN01—Establish platform velocity (in: speed, bearing; out: ack)
RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)
SFN02—Move force to a fixed position (in: coordinates; out: ack)
RPN02—Move platform to a fixed position (in: coordinates; out: ack)
RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

SFN03—Maintain force contact with a track (in: track; out: platform)
RPN03—Move platform to a track (in: track; out: ack)
RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)
SFN04—Maintain force position relative to a track (in: track; out: ack)
RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

Platform Level

SPN01—Establish platform velocity (in: speed, bearing; out: ack)
SPN02—Move platform to a fixed position (in: coordinates; out: ack)
SPN03—Move platform to a track (in: track; out: ack)
SPN04—Maintain platform position relative to a track (in: track; out: ack)
SPN05—Avoid objects (out: ack)
SPN06—Report platform position and velocity
SPN20—Report element status (in: type; out: report)

COMMUNICATION ELEMENT

Force Level

Supporting platform requests are received as commands by the platform decision support element.

SFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)
qualifiers: information type = {voice, data, record, image, etc.}
security type = {genser, SI, etc.}
reliability
delivery type = {point-to-point, multicast, etc.}
priority
start and end times
acknowledgment requirements
SFC02—Establish virtual circuit (in: qualifiers, caller address, callee addresses; out: circuit designation)
qualifiers: information type = {voice, data, record, image, etc.};
security type = {genser, SI, etc.}
reliability
delivery type = {point-to-point, multicast, etc.}
priority
start and end times
acknowledgment requirements

RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

SFC03—Disestablish virtual circuit (in: circuit designation)

RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

SFC04—limit force emissions (in: allowable platform emissions; out: ack)

RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

SFC05—Send electronic mail message (use SFC01 format)

Note: a global addressing scheme (e.g., a composite Theater/Force/Platform/User ID) is assumed

SFC06—Read electronic mail messages (in: user ID; out: messages)

RFC01—Send message (in: qualifiers, file, sender address, receiver address; out: ack)

SFC07—Transfer file (use SFC01 format)

SFC08—Remote procedure call (use SFC02 format)

SFC10—Report force emissions/comm plan (in: constraints; out: plan)

RPC10—Report platform comm plan (in: constraints; out: plan)

Platform Level

SPC02—Support virtual circuit (in: qualifiers, other addresses, circuit designation; out: ack)

SPC10—Report element emissions/comm plan (in: constraints; out: plan)

SPC20—Report element status (in: type; out: report)

Appendix F

RESOURCE ALLOCATION

Resource allocation, deallocation, and control must be priority-based and must also support deadlock prevention. These topics are discussed here.

ALLOCATION, CONTROL, AND DEALLOCATION MECHANISMS

The element controller *allocates* resources to an element. Each resource keeps its own status, controlling element name, and priority. The *status* takes values {down, available, unavailable}. The *priority* is the priority (as well as a unique identifier) of the service the resource is currently supporting for the listed *element*. If a resource is "unavailable," it is "preemptable" if the priority of the new service exceeds the current resource priority.

To support deadlock prevention (see below), resource allocation is a two-step process. The first step, in response to a query from the decision support element, is to verify that all resources required to perform a service can be allocated. For the element controller, a resource is (1) "available," (2) "unavailable but preemptable," or (3) "unavailable and not preemptable." if all necessary resources fall into categories 1 and 2, the element controller notifies the decision support element that it can perform the new service.

If any required resource is "unavailable and not preemptable" or if there are insufficient resources (among those not "down") to do the service at all, the element controller notifies the decision support element (which must wait to initiate the system service) and records the resource name and requests priority in a queue.

The second step is to allocate the resources to the newly requested service in response to a request. The element controller takes these steps: for each "available" resource, it sets the resource status to "unavailable," puts its own name in the element field, and sets the resource priority to the service priority; for each "unavailable but preemptable resource," it preempts the resource by notifying the listed element and resetting the priority to the new (higher) service priority. The element then begins the service.

If a resource being used by an element is preempted, the element must discontinue the affected service, record the resource name and request priority in a queue, deallocate any other resources currently allocated for that service, and notify the decision support element.

Whenever resources change status from "unavailable" or "down" to "available," the deallocating element or the resource itself must put a deallocation message on the

network. At that time, element controllers can reexamine resource/request queues, in priority order, to find resource requirements which can now be met.

DEADLOCK PREVENTION

Deadlock is a condition where, in a set of two or more services, each service is halted, waiting for one or more resources controlled by one or more other services in the set. The applicable BMA design rule is that *no command (system service) will be capable of causing a deadlock.* Since commands will carry different priorities, this will guarantee that no combination of commands can cause a deadlock.

One approach to deadlock prevention is to allow a system service to begin only when all necessary resources can be allocated to perform all supporting services. Since different elements and different services within elements may be counting on using the same resources, the decision support element verifies the ability of the element controller to allocate sufficient resources.

While this may appear to be an inefficient way to share resources, the temptation to settle for a weak deadlock prevention policy in the name of efficiency must be resisted; a deadlock is difficult to detect and can bring a system "to its knees." The issue of efficient use of resources can be addressed in several ways: priority-based resource preemption (already part of the architecture) assures that the highest priority system services will complete; dynamic priorities, which automatically increase with the age of the command, can be used; sharable resources reduce contention; job status reporting, timers, system-prompted user intervention, function partitioning, etc, can be used to shorten service sessions; and extensive experimentation, using models, testbeds, etc., can be used to tune the system.

IMPLEMENTING PRIORITY

The following is an approach to guarantee both higher priorities for those who need them (consistent with doctrine) and unique priorities. A three-field priority word is used where the highest order field, A, determines the range of priorities accessible to a user. For this example, assume that each user is granted a priority range commensurate with his level of authority. For example, if user 1 reports to user 2 who reports to user 3 (i.e., user 3 is highest in the chain of command), the ranges for field A available to users 1, 2, and 3 would be $\{0..A_1\}$, $\{0..A_2\}$, and $\{0..A_3\}$, respectively, where $A_1 < A_2 < A_3$. This guarantees that user 2 can always generate a command of higher priority than user 1 and that user 3 can always generate a command of higher priority than user 2.

The second highest order field, B, contains a priority discriminator. This discriminator is used, if necessary, to differentiate a new priority from an active priority at a

node. For example, suppose that command i, with a priority whose A and B fields = A_i and B_i, is active at a node. Suppose further that a user at that node wishes to create new command k having "roughly" the same level of priority, i.e., where $A = A_k = A_i$. The system must then select a value for B_k different from B_i. This approach guarantees that each command active at a single node has a unique priority.

The lowest order field of the priority word, C, contains a node ID, consisting of theater, force, and platform identifiers. Use of this field assures that no two nodes will assign identical priorities.

It will be important to establish clear policies for assigning priorities. It might be helpful to attach descriptors to priority ranges. Using the above example, assume that the OTC and the platform commanding officer have priority ranges $\{0..A_f\}$ and $\{0..A_p\}$ respectively, where $Af > A_p$. For example, guidelines to the OTC on setting the priority, A, of a command, might be the following:

- Set A between 0 and $A_p/2$ for routine commands
- Set A between $A_p/2$ and A_p for high-priority commands
- Set A between A_p and A_f *for force-critical commands.*

Appendix G

SPECIAL-PURPOSE RESOURCES

The special-purpose resources of each element are listed.

SENSING ELEMENT

Search Radars
- Air Search Radars
  - Surf Search Radars
  - Multifunction Radars
  - Electronic Support Measures Equipment

Sonars
- Active/Passive Sonars
- Active Sonars
- Passive Sonars

Electrooptical Receivers
- Infrared
- Visible light
- Ultraviolet

CommInt Sensor
IFF Receiver

ENGAGEMENT ELEMENT

Weapon Pointing Equipment
- Multifunction Radars
- Fire Control Radars

Armament
- Launchers
- Gun Mounts
- Torpedo Tubes

Ammunition
- Missiles
- Torpedoes
- Depth Charges

Active/Passive Countermeasures
- Acoustic Decoys

- Chaff
- Active Electronic Countermeasures

NAVIGATION ELEMENT

- GPS
  - GPS Receiver
  - GPS Processor
- Gyro system
- Celestial navigation system
- Compass
- Autopilot

COMMUNICATION ELEMENT

- Receivers
- Transmitters
- Antennas

Appendix H

SERVICE SESSION EXAMPLES

PLATFORM-LEVEL SENSING

The following example is intended to illustrate how the sensing element might be configured to perform a service aboard a single platform. It relates to one user request, by the platform commanding officer (CO), to initiate a service followed by a second user request to terminate that service. The example is expanded as if there were no other active user requests. In practice, many user requests would be active at any time. The sequence of events follows. The numbers in parentheses simply keep track of hardware resources.

- The CO makes user request RPS01, "surveil a volume," at the decision support element, operating on a workstation (1).
- The decision support element (1) forwards request RPS01 to the sensing element controller, running on a workstation (2).
- Upon receiving RPS01, the sensing element controller (2) allocates two sensors (3 and 4).
- The sensing element controller (2) allocates a signal processing application from a file-server (5) to run on a signal processor (6).
- The sensing element controller allocates similar-source and multisource integration algorithms from the file server (5) and track data from mass storage (7) to run on a workstation (8).
- The CO makes user-request RPS07, "terminate sensing service," at the decision support element (1).
- The decision support element (1) forwards request RPS07 to the sensing element controller (2).
- Upon receiving RPS07, the sensing element controller (2) deallocates the sensors (3 and 4).

FORCE-LEVEL COOPERATIVE ENGAGEMENT

Cooperative engagement relates to all warfare areas and levels of command (whenever offboard information is required to engage). "Silent Sam" tactics, an AAW

example, requires the shooter to remain passive while acting on other target information. In ASW/ASUW, dissimilar weapons and sensing systems may cooperate on separate platforms. Even a single surface ship may use cooperative engagement in cooperation with a LAMPS unit.

This example illustrates both the implementation of a service by the force-level engagement element and the interaction among requests and services at the force and platform levels. It assumes that the platforms and kill type have already been chosen in response to prior user requests.

The example uses three platforms: platform A hosts the OTC of a battle force; platforms B and C are selected to participate in the cooperative engagement by the OTC, who initiates the cooperative engagement via the decision support element. Platform B provides targeting information to platform C, which does the shooting. Separate sets of services and requests are spawned as a result of the single user request to perform cooperative engagement. Here, only the one user request is considered; in practice, other user requests are active at the same time. The OTC assigns a priority to the cooperative engagement user request; all services and requests generated to support that user request, at either force or platform level, inherit that priority.

The following are decision points during the service session: (1) a decision, by the force commander (OTC), to perform cooperative engagement (after consulting his situation assessment); (2) decisions by the shooter to consummate the engagement (because he has either sufficient targeting information or no more time to wait); (3) decisions by the targeting unit to pass track updates to the shooter.

Sequence of Events Aboard Platform A

- The OTC makes user-request RFE05, "initiate cooperative engagement," at the decision support element, operating on a workstation (1).
- The decision support element (1) forwards request RFE05 to the force-level engagement element controller, running on a workstation (2).
- The engagement element controller (2) then sends request RFC02, "establish virtual circuit" (between platforms B and C), to the communication element controller (3).
- The communication element controller then allocates and uses communication resources (4) to send requests RPC02, "support virtual circuit," to platforms B and C.
- Once notified by the communication element controller (3) that the virtual circuit is established, the engagement element controller (2) sends two

requests RFC01, "send message," to the communication element controller (3). The first message, for platform B, contains request RPE05 (including track ID and circuit) to "support cooperative engagement" (as tracker). The second message, for platform C, contains request RPE05 (including, weapons, kill type, and circuit) to "support cooperative engagement" (as shooter).

- The communications element sends the messages (4).
- To cancel the engagement (either because it is complete or because it is no longer useful), the OTC makes user request RFE06, "terminate cooperative engagement."
- Upon receipt of RFE06, the engagement controller (3) uses RFC01 (as above) to send request RPE06, "terminate cooperative engagement," to platforms A and B.

Sequence of Events Aboard Platform B

- The CO, at the platform decision support element (1), receives request RPC02, "support virtual circuit" with platform C, from platform A.
- The decision support element (1) forwards request RPC02 to the communications element controller (2), which allocates communication resources (3) to support the virtual circuit and returns an acknowledgment.
- The CO, at the decision support element (1), receives request RPE05 (including track ID and circuit) to "support cooperative engagement" as tracker, from platform A, and forwards it to the engagement element controller (3).
- Upon receipt of RPE05, the engagement element controller (3) makes request RPS02, "monitor a track" with surveillance quality, to the sensing element controller (4).
- Upon receipt of RPS02, the sensing element controller (4) allocates a search radar (5) and a signal-processing application from a file-server (6) to run on a signal processor (7).
- When notified by platform C (via virtual circuit), the engagement element controller (3) makes request RPS03, "monitor a track" with fire-control quality, to the sensing element controller (4).
- Upon receipt of RPS03, the sensing element controller (4) allocates a fire-control radar (8), deallocates the search radar (5), and allocates a new signal-processing application from the file server (6) to run on the signal processor (7).

- The decision support element (1) receives request RPE06, "terminate cooperative engagement," from platform A and forwards it to the engagement element controller (2).

- Upon receipt of RPE06, the engagement element controller (2) sends request RPS06, "discontinue track monitoring," to the sensing element controller (4).

- Upon receipt of RPS06, the sensing element controller (4) deallocates the signal processor (7) and the fire-control radar (8).

Sequence of Events Aboard Platform C

- The CO, at the platform decision support element (1), receives request RPC02, "support virtual circuit" with platform B, from platform A.

- The decision support element (1) forwards request RPC02 to the communications element controller (2), which allocates communication resources (3) to support the virtual circuit and returns an acknowledgment.

- The CO, at the decision support element (1), receives request RPE05 (including track ID and circuit) to "support cooperative engagement" as shooter, from platform A, and forwards it to the engagement element controller (4).

- Upon receipt of RPE05, the engagement element controller (4) demands a prelaunch application from a file-server (5) to run on a workstation (6). This application monitors the virtual circuit for tracking information from platform B to determine when a successful engagement is possible.

- Once successful engagement is possible, platform C may require confirmation from the OTC (as an acknowledgment to request RPE05). Once acknowledgment is received, the engagement element controller (4) notifies platform B, via the virtual circuit, of the need for fire-control quality tracking data. It also allocates a missile launcher (7) and a control application from the file server (5) to run on a workstation (8).

- When prelaunch is complete, and firing is to begin, the prelaunch application sends launch request RPE07 to the missile launcher (7) (via the engagement element controller) and control request RPE08 to the control application (8) (also via the controller). The engagement element controller (4) then deallocates the prelaunch application (6).

- The decision support element (1) receives request RPE06, "terminate cooperative engagement," from platform A and forwards it to the engagement element controller (4).
- Upon receipt of RPE06, the engagement element controller (4) deallocates the missile launcher (7) and the control application (8).

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1991	Interim: 1 October 90 -- 30 September 91

4. TITLE AND SUBTITLE

Battle Management Architecture Design Concepts

5. FUNDING NUMBERS

6. AUTHOR(S)

Command and Control Systems Architecture and Engineering Program Office

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Ocean Systems Center
San Diego, CA 92152-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

NOSC TR 1436

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Ocean Systems Center
San Diego, CA 92152-5000

10 SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document describes an architecture for the development, operation, and modification of battle management systems. Battle management is broader in scope than command and control, communications and intelligence (C^3I), providing the wherewithal for commanders to deploy and, if required, fight assigned forces effectively across the spectrum of conflict.

14. SUBJECT TERMS

command and control communications (C^3)
command and control, communications and intelligence (C^3I)
command and control systems
battle management
architecture
open systems
multi-warfighting
layered systems

15. NUMBER OF PAGES

79

16. PRICE CODE

17 SECURITY CLASSIFICATION OF REPORT	18 SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

21a. NAME OF RESPONSIBLE INDIVIDUAL	21b. TELEPHONE *(include Area Code)*	21c. OFFICE SYMBOL
Chuck Mirabile	(619) 553-4161	Code 405

INITIAL DISTRIBUTION

Code 0012	Patent Counsel	(1)
Code 17	R. Shearer	(1)
Code 171	J. Avery	(1)
Code 171	D. Smith	(1)
Code 171	M. Vineberg	(10)
Code 171	A. E. Chagnon	(5)
Code 40	R. C. Kolb	(2)
Code 402	B. Wasilausky	(1)
Code 402	J. Maynard	(1)
Code 405	C. Mirabile	(20)
Code 405	I. Sabalillah	(1)
Code 405	V. Monteleon	(5)
Code 405	J. Kennedy	(2)
Code 405	P. Garcia	(1)
Code 405	S. Hearold	(1)
Code 41	A. Justice	(1)
Code 41	M. Crowley	(5)
Code 411	G. H. Schulte	(1)
Code 412	L. J. Core	(1)
Code 413	M. Butterbrodt	(1)
Code 413	G. Myers	(20)
Code 414	A. D. Gomez	(1)
Code 42	J. A. Salzmann	(1)
Code 421	D. L. Conwell	(1)
Code 421	F. White	(5)
Code 422	D. K. Porter	(1)
Code 423	R. E. Pierson	(1)
Code 424	J. M. Chevrier	(1)
Code 425	S. B. Schneiderman	(1)
Code 43	G. F. Changler	(1)
Code 431	D. G. Mudd	(1)
Code 432	T. R. Tiernan	(1)
Code 433	J. R. Beauchane	(1)
Code 434	R. M. Akita	(1)
Code 435	D. C. Lutz	(1)
Code 44	J. D. Grossman	(1)
Code 44	L. Duffy	(2)
Code 441	C. M. Dean	(1)
Code 442	C. E. Englund	(1)
Code 442	B. Feher	(2)
Code 443	E. A. Koehler	(1)
Code 444	D. C. Eddington	(1)
Code 45	H. F. Wong	(1)
Code 451	R. Hollandsworth	(1)
Code 452	R. F. Smith	(1)
Code 453	K. G. Kaufmann	(1)
Code 454	T. M. Fitzgerald	(1)
Code 455	J. Kadane	(1)
Code 46	R. B. Volker	(1)
Code 461	R. E. Laverty	(1)
Code 462	A. D. Sandlin	(1)

INITIAL DISTRIBUTION (Cont'd)

Code 463	R. H. Kolb, Jr.	(1)
Code 464	C. H. Sturtevant	(1)
Code 465	T. H. Lockhart	(1)
Code 604	G. Carlson	(10)
Code 624	D. Mattison	(5)
Code 70	E. Shutters	(1)
Code 753	H. Quesnell	(1)
Code 80	K. Regan	(1)
Code 82	P. Adams	(1)
Code 82	R. Kochanski	(1)
Code 821	B. Reed	(1)
Code 952B	J. Puleo	(1)
Code 961	Archive/Stock	(6)
Code 964B	Library	(3)

Defense Technical Information Center
Alexandria, VA 22304-6145 (4)

NOSC Liaison Office
Washington, DC 20363-5100

Navy Acquisition, Research & Development Information Center (NARDIC)
Alexandria, VA 22333

David Taylor Research Center
Bethesda, MD 20084-5000

Naval Sea Systems Command
Washington, DC 20362-5101

Naval Research Laboratory
Washington, DC 20375-5000 (2)

Naval Weapons Center
China Lake, CA 93555-6001 (2)

Naval Surface Warfare Center
Silver Spring, MD 20910 (2)

Naval Coastal Systems Center
Panama City, FL 32407-5000 (3)

United States Coast Guard
Washington, DC 20593

Johns Hopkins University
Applied Physics Laboratory
Laurel, MD 20707-6090 (2)

Systems Exploration, Inc.
San Diego, CA 92117

OASD (P/L) WSIG
Washington, DC 20301-8000

Center for Naval Analyses
Alexandria, VA 22302-0268

Office of Naval Technology
Arlington, VA 22217-5000

David Taylor Research Center
Annapolis, MD 21402-5067 (2)

Naval Ship Systems Engineering Station
Philadelphia, PA 19112-5083

Naval Air Development Center
Warminster, PA 18974-5000

Naval Surface Warfare Center
Dahlgren, VA 22448 (2)

Naval Underwater Systems Center
Newport, RI 02841

Space & Naval Warfare Systems Command
Washington, DC 20363-5100 (41)

Naval Postgraduate School
Monterey, CA 93943

Advanced Technology & Research Corporation
Laurel, MD 20707